Computers and Operations Research 83 (2017) 78–94

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A fast two-level variable neighborhood search for the clustered vehicle routing problem

Christof Defryn*, Kenneth Sörensen

University of Antwerp, Departement of Engineering Management, ANT/OR - Operations Research Group, Belgium

ARTICLE INFO

Article history:
Received 11 May 2016
Revised 24 November 2016
Accepted 8 February 2017
Available online 9 February 2017

Keywords:
Clustered vehicle routing problem
Variable neighborhood search
Metaheuristic

ABSTRACT

In this paper, we present an improved two-level heuristic to solve the clustered vehicle routing problem (CluVRP). The CluVRP is a generalization of the classical capacitated vehicle routing problem (CVRP) in which customers are grouped into predefined clusters, and all customers in a cluster must be served consecutively by the same vehicle. This paper contributes to the literature in the following ways: (i) new upper bounds are presented for multiple benchmark instances, (ii) good heuristic solutions are provided in much smaller computing times than existing approaches, (iii) the CluVRP is reduced to its cluster level without assuming Euclidean coordinates or distances, and (iv) a new variant of the CluVRP, the CluVRP *with weak cluster constraints*, is introduced. In this variant, clusters are allocated to vehicles in their entirety, but all corresponding customers can be visited by the vehicle in any order.

The proposed heuristic solves the CluVRP by combining two variable neighborhood search algorithms, that explore the solution space at the cluster level and the individual customer level respectively. The algorithm is tested on different benchmark instances from the literature with up to 484 nodes, obtaining high quality solutions while requiring only a limited calculation time.

1. Research context and literature review

Introduced by Dantzig and Ramser (1959), the vehicle routing problem (VRP) is one of the best known and most widely studied problems in the Operations Research community. Many variants of the VRP have been proposed and solved during the last decades. In this paper, we focus on the *clustered vehicle routing problem* (CluVRP), a variant of the capacitated vehicle routing problem (CVRP) in which all customers are partitioned into predefined *clusters*. In the *strict* version of the CluVRP, all customers belonging to the same cluster should be visited by the same vehicle consecutively in the same path. In other words, when a customer is visited by a vehicle, all other customers belonging to the same cluster should be visited first before the vehicle can either return to the depot or move to a client that belongs to another cluster. We refer to this problem as the *clustered vehicle routing problem with strong cluster constraints*. In Section 5, we will define a new variant of the problem with *weak cluster constraints*.

The idea of customer clustering was introduced by Chisman (1975) when defining the *clustered travelling salesman problem* (CluTSP). The objective of this problem is to construct a Hamiltonian path with minimum distance, visiting all customers exactly once. Customers, however, are assigned to a set of predefined clusters and an extra constraint imposes that all customers belonging to the same cluster should be served consecutively. The main algorithmic contributions regarding the CluTSP consist of a tabu search heuristic (Laporte et al., 1997), genetic algorithms (Ding et al., 2007; Potvin and Guertin, 1996) and a path relinking approach including GRASP (Mestria et al., 2013). In addition to a number of vehicle routing applications, the CluTSP can also be applied in many other fields, 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) (Laporte et al., 2002).

The CluVRP was introduced by Sevaux and Sörensen (2008) in order to model the parcel delivery activities of courier companies. A common practice in this industry is to sort all outbound parcels into bins, where each bin corresponds to a specific, predefined part of the distribution area, called a *zone*. The first step in solving the distribution planning problem of a courier company is to assign these bins (zones) to the vehicles available. A multi-objective approach for this problem is presented by Janssens et al. (2015). Afterwards, an optimal cluster and customer sequence should be determined for every vehicle. Other examples involving customer clustering can be found in situations where it is desirable that certain customers are served by the same vehicle. This might be due to the fact that some customers demand a similar service, request

* Corresponding author.
E-mail address: christof.defryn@uantwerpen.be (C. Defryn).

http://dx.doi.org/10.1016/j.cor.2017.02.007
0305-0548/

a specific repairman skill, or if the customer-driver relationship is perceived important by one of the parties.

In Pop et al. (2012), an exact method for solving the CluVRP is developed as an extension of the Generalized Vehicle Routing Problem (GVRP). The GVRP is closely related to the CluVRP, as both problems share the concept of customer clustering. Contrary to the CluVRP, the GVRP requires that only one customer is visited in every cluster (Ghiani and Improta, 2000). A new compact and effective integer programming formulation and exact solution approach is proposed in Battarra et al. (2014).

To solve the CluVRP heuristically, Barthélemy et al. (2010) introduce a transformation of the CluVRP into the CVRP. This is done by adding a large distance M to all inter-cluster edges in the distance matrix. As a result, routes are obtained in which all customers of a single cluster are served before leaving the cluster, because of the high penalty costs. In Barthélemy et al. (2010) this *big M approach* is further combined with a simulated annealing heuristic.

A hybrid algorithm that does not make use of the big M transformation is proposed in Marc et al. (2015), but this algorithm makes use of precomputed cluster centers and is therefore only able to solve Euclidean instances. Furthermore, no calculation times are mentioned.

Two alternative metaheuristic solution approaches are proposed by Vidal et al. (2015). The first one is an adaptation of the Iterated Local Search (ILS) algorithm developed by Subramanian (2012) for the CVRP. In order to avoid the evaluation of many infeasible moves, due to the additional cluster constraints, the neighborhoods are redefined. Secondly, Vidal et al. (2015) use their Unified Hybrid Genetic Search (UHGS) approach to solve the CluVRP. Since this method is designed to solve the non-clustered VRP, the precomputation of all intra cluster Hamiltonian paths is required. The authors report high quality solutions for both methods. This solution quality, however, comes at the expense of very high calculation times.

Defryn and Sörensen (2015) propose a decomposition of the problem in two optimisation levels: a high-level routing problem at the cluster level and a low-level routing problem at the individual customer level. Expósito-Izquierdo et al. (2016) acknowledge the two-level optimisation strategy and propose a solution algorithm that combines the Record-to-record algorithm (Li et al., 2007) at the cluster level with the Lin–Kernighan heuristic (Lin and Kernighan, 1973) to determine the intra-cluster routes.

The current paper contributes to the existing literature in the following ways. First, we are able to report improved results on most of the instances provided by Expósito-Izquierdo et al. (2016). Secondly, even though good algorithms exist for solving the CluVRP, most notably the ones proposed by Vidal et al. (2015), a gap remains for an approach that allows to calculate good solutions in a *short amount of computing time*. The heuristic procedure proposed in this paper is able to generate good quality feasible solutions very fast. Such an algorithm is necessary in situations where large calculation times are not available or impractical, such as in the daily planning process of couriers or other transportation companies. It is additionally useful in applications for which the CluVRP is solved many times as a subproblem. For example, the problem of defining the optimal customer clusters in the distribution area will rely on the CluVRP solution as an evaluation criterion. In this case, a fast evaluation is preferred over the fact that the optimal solution is guaranteed. A third contribution is that, compared to Defryn and Sörensen (2015) and Expósito-Izquierdo et al. (2016), we generalise the two-level framework to also handle non-Euclidean instances. Finally, a new CluVRP variant, i.e., the CluVRP *with weak cluster constraints* is introduced in this paper. For some applications, the use of clusters might be beneficial to some extent (e.g., the sorting of the packages and the allocation of zones to vehicles for courier companies, as described above), but could be relaxed when it comes to optimising the route of a single vehicle. In other words, the CluVRP with *weak* cluster constraints still enforces that all customers belonging to the same cluster are visited by the same vehicle, but relaxes the constraint that they should be visited *consecutively*. The customers in the clusters assigned to a vehicle therefore can be visited in any order. To the best of our knowledge, this problem has not yet been described in the literature.

The structure of the paper is as follows. In Section 2, the CluVRP is formally described, after which a detailed analysis of the developed metaheuristic is performed in Section 3. Our algorithm is tuned and tested on multiple instances of different sizes in Section 4. The CluVRP with weak cluster constraints is introduced and compared to the original strong cluster constraint variant in Section 5. Finally, the main conclusions are summarised in Section 6.

2. Problem definition

In the CluVRP with strong cluster constraints, we are given a complete undirected graph $G = (V, E)$, where V is a set of vertices including one depot (denoted as V_0) and multiple customer nodes. A distance d_{ij}, is associated with each edge $(i, j) \in E$ connecting two nodes. We consider K to be a set of homogeneous vehicles with a maximum capacity Q each. All vehicles start and end their trip at the depot. For each customer i the demand is denoted by q_i. Furthermore, a set of clusters is denoted by R. Cluster $r_0 \in R$ only contains one node, the depot. All other clusters contain at least one customer. The set of customers in a cluster is denoted as $C_r = \{i \in V\backslash V_0 : r_i = r\}$, $\forall r \in R$.

Following Expósito-Izquierdo et al. (2016), the CluVRP can be defined by the mathematical model described below which requires the definition of some additional variables. Consider Z to be any subset of V that is different from V. Then, let $\delta^+(Z)$ be the set of edges $(i, j) \in Z \times V\backslash Z$ and $\delta^-(Z)$ the set of edges $(i, j) \in V\backslash Z \times Z$.

$$x_{ijk} = \begin{cases} 1 & \textit{vehicle k travels from node i to node j} \\ 0 & \textit{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \textit{customer i is served by vehicle k} \\ 0 & \textit{otherwise} \end{cases}$$

$$\min \sum_{(i,j)\in E} \sum_{k\in K} d_{ij} x_{ijk} \tag{1}$$

Subject to

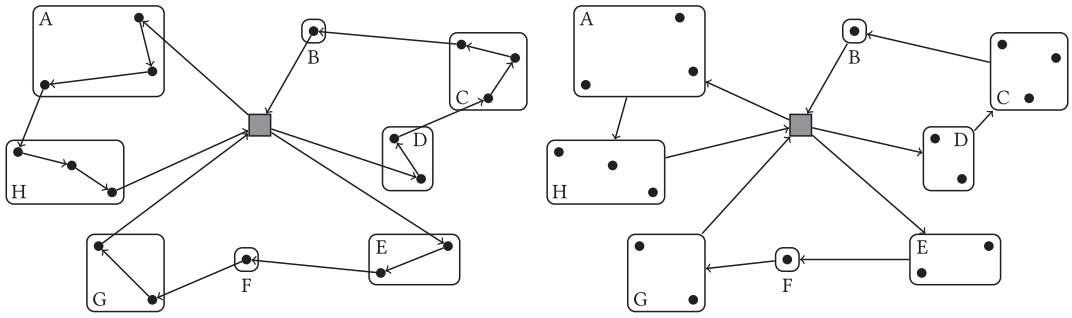
$$\sum_{k\in K} y_{ik} = 1 \qquad \forall i \in V\backslash V_0 \tag{2}$$

$$\sum_{k\in K} y_{0k} = |K| \tag{3}$$

$$\sum_{j\in V\backslash V_0} x_{ijk} = \sum_{j\in V\backslash V_0} x_{jik} = y_{ik} \qquad \forall k \in K, \forall i \in V \tag{4}$$

$$\sum_{i\in V} q_i y_{ik} \le Q \qquad \forall k \in K \tag{5}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijk} \le y_{hk} \qquad \forall Z \subseteq V\backslash V_0, \forall h \in Z, \forall k \in K \tag{6}$$



(a) A CluVRP solution at the customer level *(b) A CluVRP solution at the cluster level*

Fig. 1. The CluVRP with strong cluster constraints. All three vehicles depart from the central depot (gray rectangle) to serve all the customers (black circles). All customers within the same cluster should be served consecutively by the same vehicle.

$$\sum_{(i,j)\in\delta^+(C_r)}\sum_{k\in K} x_{ijk} = \sum_{(i,j)\in\delta^-(C_r)}\sum_{k\in K} x_{ijk} = 1 \qquad \forall r \in R \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K \tag{8}$$

$$y_i \in \{0,1\} \qquad \forall i \in V, \forall k \in K \tag{9}$$

In the model formulation above, the objective function (1) minimises the total distance travelled by all vehicles. Constraints (2) ensure that each customer is visited exactly once. Contraints (3) state that all vehicles should visit the depot. Constraints (4) guarantee that the same vehicle that arrives at a customer also leaves from that customer. Constraints (5) make sure that vehicle capacities are respected. The subtour elimination constraints are represented by Eq. (6). Constraints (7) establish that each cluster is visited exactly once by one vehicle.

The CluVRP is visualised in Fig. 1. On the left hand side, the final solution of the CluVRP with strong cluster constraints is shown at the individual customer level. The corresponding solution at the cluster level is included at the right hand side. This high level representation will be used during the algorithm to reduce the complexity of the problem by exploiting its clustered substructure.

As demonstrated by Lenstra and Kan (1981), the CVRP is NP-hard. Since any CVRP can be reduced to a CluVRP with one customer in each cluster and 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

We propose a metaheuristic approach that explores the solution space at two different levels: the *cluster level* and the *customer level*. At both levels, a Variable Neighborhood Search (VNS) algorithm is used to find a local optimum. VNS, introduced by Mladenović and Hansen (1997) has proven to be a successful framework for solving combinatorial optimisation problems, especially vehicle routing problems (Hansen and Mladenović, 2014). First, the problem is solved at the cluster level. Afterwards, this result is used as an input for the customer level VNS. During the diversification phase, the algorithm moves back from the customer to the cluster level. The outline of our heuristic is shown in Algorithm 1. In the following sections, we take a closer look at the different operators and their implementation.

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

```
1: nIterationsNoImprovement ← 0;
2: goToNodeVNS ← false; stoppingCriterion ← false;
3: best solution found: S_i^* = ;
4: objective value of best solution found: f(S_i^*) ← ∞ ;
5: Step 0: Precomputation
6: calculate-inter-cluster-distances();
7: Step 1: Constructive phase
8: S_c ← allocate-clusters-to-vehicle();
9: Step 2: Intensification phase
10: do
11:     S_c' ← perform-VNS-at-cluster-level(S_c);
12:     do
13:         S_i ← convert-from-cluster-to-customer-level(S_c');
14:         S_i' ← perform-VNS-at-customer-level(S_i);
15:         if f(S_i') < f(S_i^*) then
16:             S_i^* ← S_i';
17:             f(S_i^*) ← f(S_i');
18:             nIterationsNoImprovement ← 0;
19:         else
20:             nIterationsNoImprovement ← nIterationsNoImprovement + 1;
21:             if nIterationsNoImprovement = maxIterationsNoImprovement
    then
22:                 stoppingCriterion ← true;
23:                 break;
24:             end if
25:         end if
26:         Step 3: Diversification phase
27:         S_c ← perturb(S_c');
28:         repair(S_c);
29:         r ← get-random-number[0,1]();
30:         if r < cluVNSProb then
31:             goToNodeVNS ← false;
32:         else
33:             goToNodeVNS ← true;
34:         end if
35:     while goToNodeVNS = true;
36: while stoppingCriterion = false;
37: return S_i^*;
```

3.1. Precomputation

During precomputation all inter cluster distances are quantified. As described earlier, only the distance $d_N(i, j)$ between individual nodes i and j is given. These distances are not necessarily symmetrical or Euclidean. To solve the CluVRP at the cluster level (i.e., to determine the assignment of clusters to vehicles and the sequencing of the clusters per vehicle), the *inter cluster distance matrix* should be defined. For this purpose we use the shortest edge between two clusters as an approximation for the inter-cluster distance as this will be the preferred edge to go from one cluster to another in the low-level routing solution.

3.2. Constructive phase

The main goal of the constructive phase is to generate a feasible initial solution at the *cluster level*. This means that for every cluster, the individual customers are disregarded and the cluster as a whole is allocated to an available vehicle. Even though the travel times between the clusters are taken into account during the constructive process, constructing a feasible allocation of clusters to vehicles is the priority in this phase. Therefore, instead of using a VRP algorithm, a *bin packing approach* is preferred here. This design choice is justified as follows.

First, contrary to a standard VRP formulation, the number of vehicles is given and should not be optimised any further, as all vehicles must be used anyway. As a result, the exact number of trips is known in advance. Furthermore, it can be argued that the inter cluster Hausdorff distances are only an approximation of the distances between two clusters, as the real distance depends on both the cluster and customer sequence in the trip. Finally, the instances (especially the smaller ones) are constructed in such a way that only very little to no spare capacity is available. For these reasons, a problem specific bin packing approach is more suitable here.

If we disregard the travel of the vehicles between clusters, the allocation of clusters with their given demand to a set of vehicle can be modeled as a *one-dimensional bin packing problem with given number of bins*. A set of items (clusters) with a given weight (total demand) are to be packed into a set of bins (vehicles) with a predefined maximal load (vehicle capacity Q).

The one-dimensional bin packing problem is shown to be strongly NP-complete (Garey and Johnson, 1978). Because we are not interested in the optimal bin packing solution, but a solution for the CluVRP, we prefer a fast algorithm that provides us with a feasible result. 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 adopted.

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 satisfactory, but when solving the CluVRP, it is important that efficient routes can be constructed afterwards with the clusters that have been allocated to the same vehicle. For every cluster, sorted in decreasing order according to demand, we therefore look at the latest cluster that was added to all the bins (vehicles). We then prefer the vehicle for which this latest cluster is located the closest to the current cluster to add. 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 (certain clusters are assigned to that vehicle), we force that other clusters in that same direction are also allocated to this vehicle.

Due to the deterministic character of this constructive heuristic, the same initial solution will be generated during every run of the algorithm. In order to prevent this from happening, an alternative strategy, which involves some randomness, is defined. With a predefined probability randConstructProb, the current cluster is not allocated to the closest vehicle, but from all feasible vehicles (vehicles with enough spare capacity) one vehicle is selected at random.

3.3. Redistribution algorithm

As for each instance the number of vehicles is given, the heuristic constructive procedure used to allocate clusters to vehicles might reach a point where no vehicle has enough capacity left to store the next cluster. In order to cope with these situations, a specific redistribution operator, that tries to re-optimize the current capacity distribution, is built into the solution algorithm.

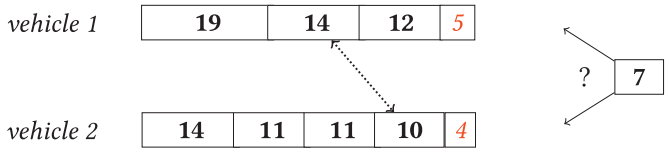


Fig. 2. Visualisation of the *clusterSwap* operator. By swapping the clusters with demand of 14 and 10, free space is created in vehicle 1 where the additional cluster with a demand of 7 can be placed.

This redistribution algorithm aims to increase the fill rate of one of the vehicles as much as possible by means of a clusterSwap operator, that swaps the vehicle of two already allocated clusters. All cluster pairs are checked sequentially, and the best move — the one leading to the highest possible fill rate for one of the vehicles — is executed.

The clusterSwap operator is illustrated in Fig. 2. Two vehicles are considered with a capacity of 50 items each and eight clusters with a demand of 19,14,14,12,11,11,10 and 7 items should be allocated to one of the vehicles. As shown, adding the last cluster to any of the vehicles will result in an infeasible solution. This issue is solved by swapping the vehicles assigned to the clusters with a demand of 14 and 10, which will result in a 100% utilisation of vehicle 2. As a results, enough spare capacity becomes available in vehicle 1 to hold the cluster with a demand of 7.

3.4. Intensification phase

From the moment that a feasible solution is constructed, the algorithm starts the intensification phase in which the initial solution is improved until a local optimum is reached. This is done in three steps.

First, the initial solution is improved at its cluster level by means of a VNS. A locally optimal cluster sequence is obtained for every vehicle. Afterwards, this cluster level solution is translated to the individual customer level by a conversion operator. The obtained result are then used as the input for a second VNS at the individual customer level.

3.4.1. Intensification at the cluster level

The first part of the intensification phase is executed at the level of the clusters and uses the inter cluster distance matrix, constructed during precomputation. By ignoring all individual customer nodes, the problem size and complexity are reduced. The obtained high level routing problem is solved by a VNS with the objective of finding an optimal cluster sequence for every vehicle. The search is based on seven local search operators that are commonly used in vehicle routing. Both *intra and inter vehicle neighborhoods* are explored. The intra vehicle operators try to minimize the total distance of a single trip. The inter vehicle operators combine at least two trips while trying to improve the global cost (total distance) of the solution by moving one or exchanging a set of clusters between different vehicles. All local search operators are shown in Table 1 and have complexity $O(n^2)$. We use the *first improvement strategy*, as every beneficial move encountered is executed by the algorithm.

The order in which the neighborhoods are checked by the algorithm is changed randomly each time the VNS is called. When no improvement can be found in the current neighborhood, the algorithm moves to the next neighborhood. Every time an improvement is found, the algorithm returns to the first neighborhood. This is repeated until none of the neighborhoods is able to improve the current solution any further, and a local optimum is reached at the cluster level.

Table 1
List of all intra and inter vehicle local search operators, implemented in the VNS at the 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 trip.
Two-Opt	Remove two edges and replace them by two new edges to close the tour.
Or-Opt	Remove N consecutive clusters and insert them at a different position in the trip. (with $N = \{2, 3, 4\}$)
Inter vehicle operators	
Swap	Swap the vehicle of two clusters.
Relocate	Remove one cluster and insert it in another trip.
Or-Opt	Remove N consecutive clusters and insert them in another trip. (with $N = \{2, 3, 4\}$)

3.4.2. Conversion operator

The best cluster sequence for every vehicle obtained during the intensification phase at the cluster level is converted into a solution at the customer level before sending it to the customer level intensification phase. This is done by the conversion operator.

For each cluster, an intra cluster TSP was constructed heuristically during pre-processing. The order in which the nodes appear in this TSP is maintained by the conversion operator. The starting node is chosen as the node closest to the current position of the vehicle. In other words, when entering a new cluster, the customer that is closest to the vehicles' current position (either the last customer visited in the previous cluster of the depot) is visited first. Starting from this customer, the node sequence equals the intra cluster TSP constructed during pre-processing.

With a probability given by the parameter randConversionProb, all nodes of the current clusters are added randomly to the solution. In this way, we introduce some diversification in the conversion operator and a larger part of the solution space is searched.

The solution obtained after applying the conversion operator is considered the initial solution at the customer level.

3.4.3. Intensification at the customer level

The initial solution at the customer level constructed by the conversion operator, is improved further during a second intensification phase in which all individual customer nodes are taken into account. Similar to the VNS discussed in Section 3.4.1, a set of neighborhoods is explored in the search for a local optimal solution.

The cluster constraints, however, impose that all customers belonging to the same cluster should remain visited consecutively in the same path. This restricts the number of feasible moves to be checked by the local search operators.

Two main groups of neighborhoods can be distinguished: the *intra cluster* and the *inter cluster* neighborhoods. The first group is responsible for improving the Hamiltonian path within a certain cluster. 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 neighborhoods operate on the cluster order as obtained by the VNS at the cluster level. As no customer information was taken into account at the 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 belonging to different vehicles are swapped). The neighborhoods used by the VNS at the customer level are described in Table 2.

Similar to the first VNS, the applied neighborhoods are checked sequentially. When an improvement is found, the algorithm restarts by exploring the first neighborhood. It continues until none of the neighborhoods is able to improve the solution any further and a local optimum is reached. Again, the order in which the neighborhoods are checked by the algorithm is changed randomly by each call of the VNS.

3.5. Diversification phase

After having evaluated the new solution obtained at the customer level, the algorithm executes its diversification strategy to continue the search and explore another part of the solution space. This diversification operator consists of a *perturbation operator*, that destroys parts of the solution, followed by a *repair operator*.

As all customer nodes that belong to the same cluster should remain grouped together, the removal of one customer node by the perturbation operator would in the end result in the removal of the complete cluster this customer belongs to. Therefore, the perturbation operator is applied immediately to the current solution at the cluster level. A random part of the solution, denoted by the parameter pertRate, is destroyed and the removed clusters are stacked in a separate list.

Afterwards, all removed clusters are reallocated to random vehicles by the repair operator while making sure that the vehicle capacity constraints are not violated. If no feasible vehicle can be found for a certain cluster, the redistribution operator (described in Section 3.3) is called by the algorithm.

When a new solution is obtained, the algorithm can resume its search at two different points. Either the new solution at the cluster level is improved first by the intensification phase at the cluster level, or the algorithm calls the conversion operator immediately after the diversification phase. The probability to call the VNS at the cluster level after the diversification operator is denoted by the parameter cluVNSprob.

4. Parameter tuning and experimental results

4.1. Parameter tuning

The algorithm is controlled by four parameters, summarised in Table 3. In order to fully test the impact of these parameter values on the solution quality, a full factorial statistical experiment is executed. The tuning is done on a selection of large-size instances as provided by Battarra et al. (2014). See Section 4.3 for a more elaborate presentation of the instances. The results of this tuning procedure are visualised in Fig. 3, for each individual parameter. The obtained best parameter settings are shown in the last column of Table 3. The optimal value for the randConstructProb parameter is zero, stating that allocating each cluster to a close-by vehicle outperforms a randomized approach. For the cluVNSprob parameter the optimal value equals 1, meaning that after the diversification phase the best strategy is again to first optimise the solution at the cluster level. All results in this paper are obtained using these optimal parameter values.

4.2. Stopping criterion

The search procedure continues until a stopping criterion is reached. Our algorithm uses a *predefined number of iterations without improvement*. It can be expected that the solution quality will increase for a larger number of allowed iterations. This result is shown in Fig. 4. This graph is constructed by using the optimal parameter settings, defined above, while varying the maximum number of iterations without improvement. As expected, we find an almost linear relationship between the calculation time and the number of iterations without improvement. A large improvement of the solution quality is realised during the first seconds of execution. As the optimality gap decreases, more calculation time is re-

Table 2
The different VNS neighborhoods at the customer level.

Intra cluster operators	
Swap	Swap the position of two customers within the same cluster in a single trip.
Relocate	Remove one customer 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 consecutive customers and insert them at a different position within the same cluster in the trip. (with $N = \{2, 3, 4\}$)
Inter cluster operators *(intra vehicle)*	
Swap	Swap the position of two clusters within the same trip.
Relocate	Remove all customers of a single cluster and insert them sequentially at a different position in the same trip.
Inter cluster operators *(inter vehicle)*	
Swap	Swap the vehicle of two clusters.
Relocate	Remove all customers of a single cluster and insert them sequentially in another trip.

Table 3
Results of parameter tuning on a small subset of the large-size instances. These results are obtained by performing a full-factorial statistical experiment.

Parameter	Definition	Tested values	#	Best
randConstructProb	Probability that a cluster is allocated to a random instead of the closest feasible vehicle during construction.	0,0.1,...,0.5	6	**0**
randConversionProb	Probability that an intra cluster route is inserted randomly instead of using the nearest neighbour approach.	0,0.1,...,1	11	**0.4**
pertRate	Percentage of the solution that is randomly destroyed by the perturbation operator.	0,0.1,...,0.5	6	**0.1**
cluVNSprob	Probability that, after the diversification phase, the new solution is improved first at the cluster level before going to the conversion operator.	0.2,0.3...,1	9	**1**

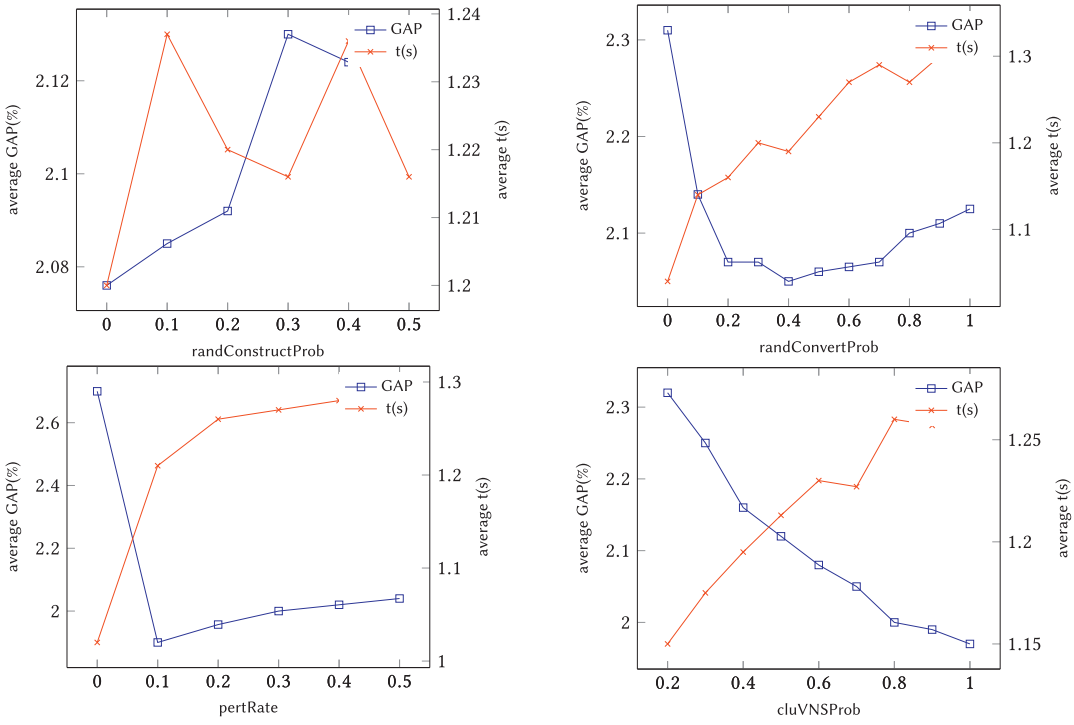


Fig. 3. Solution quality, measured by the average optimality gap and the average calculation time, for different parameter settings. Results are obtained by performing a full factorial experiment.

quired to further improve the current solution. To preserve a good balance between computation time and solution quality, the stopping criterion is set to 5000 iterations without improvement.

4.3. Experimental results

The metaheuristic is tested extensively on benchmark instances of different sizes and with varying degree of clustering. All computational results are obtained using an Intel(R) Core(TM) i7-4790 @ 3.60 GHz with 16GB of RAM. Because the algorithm makes use of randomness during the optimisation process, the instances are solved multiple times (20 runs per instance). Both the average and best results are reported per instance. For a complete overview of all obtained solutions, we refer to Appendix A.

4.3.1. Results on the GVRPθ3 instances

The algorithm is tested on a set of 79 small and medium sized test instances, denoted as GVRPθ3, provided by Battarra et al. (2014), as discussed in their paper on *exact algorithms for the Clustered Vehicle Routing Problem*. These GVRP instances are adaptations of existing CVRP instances from Bektas et al. (2011). The transformation is achieved by creating clusters of customers using a seed-

Table 4
Results for the GVRPθ3 instances. Comparison between the branch-and-cut and price (BCP), branch-and-cut (BC) (Battarra et al., 2014) and the two-level VNS proposed in this paper.

		BCP		BC		VNS (20 runs)			
		Opt.	Avg. t(s)	Opt.	Avg. t(s)	Opt.	Avg. t(s)	Avg. best GAP	Avg. GAP
A	31–79 cust.	27/27	42.52	27/27	4.84	24/27	0.05	0.07%	0.07%
B	30–77 cust.	23/23	7.69	23/23	4.99	21/23	0.04	0.03%	0.04%
P	15–100 cust.	24/24	0.48	24/24	3.77	23/24	0.06	0.00%	0.02%
M+G	100–261 cust.	2/5	157.25	3/5	25.44	3/5	5.98	0.04%	0.18%
total		76/79		77/79		71/79			
average			26.87		5.86		0.43	**0.03%**	0.04%

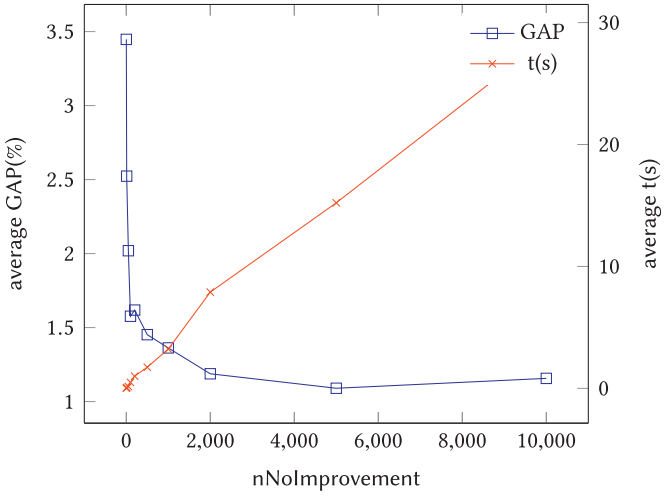


Fig. 4. Relationship between the number of iterations without improvement (stopping criterion) and the average optimality gap for the tuning instance set.

based algorithm and by replacing the number of required vehicles by the solution (number of bins) of the bin packing problem for each instance. See Battarra et al. (2014) for a more elaborate explanation on the design of the instances. The resulting number of clusters reach 88, while the average number of customers per cluster is three.

The obtained results for the GVRPθ3 instances are summarised in Table 4. By using their branch-and-cut method, Battarra et al. (2014) are able to solve 77 out of 79 instances exactly within reasonable time limits. It should be mentioned that for these methods the preprocessing times, which lie between 3 and 8 s, are not included in the calculation times. 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 the customer level for a given sequence of clusters. Our VNS algorithm is able to solve 71 instances to optimality, while significantly reducing calculation times. For the other instances, we are able to provide a high quality solution that lies on average 0.04% from optimality. For the instance G-n262-k25-C88-V9 which could not be solved using the exact approach, a heuristic solution with an objective value of 3310 is obtained in a very short calculation time. For instance M-n200-k16-C67-V6 we improve the upper bound to 909.

4.3.2. Results on the adapted Golden instances proposed in Expósito-Izquierdo et al. (2016)

In this section we test our algorithm on the CluVRP benchmark instance sets proposed by Expósito-Izquierdo et al. (2016). The instances are adaptations of the instances introduced by Golden et al. (1998) for the CVRP. Each set is characterised by a parameter ρ, representing the filling range of a vehicle. When $\rho = 100\%$, each vehicle can serve at most one cluster, whereas a lower filling percentage indicates that a higher number of clusters can be combined in one vehicle trip. Five different instance sets are built by setting $\rho \in \{10, 25, 50, 75, 100\}$%.

The two-level solution approach from Expósito-Izquierdo et al. (2016) is compared to the two-level VNS introduced in this paper. Although both algorithms are based on a breakdown of the CluVRP in two distinct routing problems, some important differences can be identified. First, Expósito-Izquierdo et al. (2016) define the high-level routing problem by replacing all clusters by their *virtual center* by using the *center of mass concept*. Their approach is therefore limited to Euclidean instances. Secondly, the algorithm by Expósito-Izquierdo et al. (2016) is mainly focused on optimising the solution at the cluster level in which the individual customer sequence is only considered in the Lin-Kernighan heuristic. Our two-level VNS, however, also includes a strong intensification of the low-level routing problem by means of a separate VNS procedure. The results below indicate that it is beneficial to devote additional attention to the solution at the customer level.

The results of both approaches are summarised in Tables 5–9. To allow a fair comparison, a maximum calculation time of 60s is considered for both algorithms. It can be seen that if $\rho \in \{10, 25, 50\}$%, almost all instances can be improved by our two-level VNS algorithm compared to the results of Expósito-Izquierdo et al. (2016). The reduction in total cost can be up to 6.10% (instance 4, with $\rho = 10\%$). For the instance sets with a higher filling rate ($\rho \in \{75, 100\}$) it turns out to be harder to improve the current best solution. This might be due to the fact that the lower the number of clusters inside a vehicle, the more the CluVRP converges to the more traditional CVRP. Therefore we lose the advantage of exploiting the clustered structure of the problem in our algorithm. However, the gaps remain far below 1% compared to Expósito-Izquierdo et al. (2016) for all instances, ensuring that a high quality and competitive solution is found.

4.3.3. Results on the adapted Golden instances proposed in Vidal et al. (2015)

Finally, we test our algorithm on yet another adaptation of the Golden et al. (1998) instance set, proposed by Battarra et al. (2014). We refer to Table 10 for an overview of the obtained results. Our two-level VNS is compared to the solution procedure of Expósito-Izquierdo et al. (2016), described above, and the UHGS algorithm from Vidal et al. (2015). The UHGS algorithm aims at combining the diversification strength of a genetic algorithm with the improvement capabilities of local search and has proven to return high quality solutions that are very close to optimality. The algorithm relies, however, on the exact solution of all intra cluster Hamiltonian paths, precomputed by means of Concorde (Applegate et al., 2006). This causes the required calculation time to increase up to even above the exact solution approach of Battarra et al. (2014). Although in terms of solution quality the UHGS method tends to outperform all existing approaches, the high calculation

Table 5
Results for the adapted problem instances by Golden et al. (1998) with $\rho = 10\%$. Comparison between the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper.

Index	n+1	Q	Best known	Expósito-Izquierdo et al. (2016)	Two-level VNS	Gap
1	240	550	**5759.25**	5801.67	**5759.25**	−0.73%
2	320	700	**9247.92**	9649.67	**9247.92**	−4.16%
3	400	900	**12904.60**	13249.22	**12904.60**	−2.60%
4	480	1000	**17810.40**	18966.92	**17810.40**	−6.10%
5	200	900	**8960.31**	9479.74	**8960.31**	−5.48%
6	280	900	**10976.50**	11601.77	**10976.50**	−5.39%
7	360	900	**12485.80**	13243.13	**12485.80**	−5.72%
8	440	900	**13331.20**	13756.51	**13331.20**	−3.09%
9	255	1000	**710.64**	717.16	**710.64**	−0.91%
10	323	1000	**908.89**	914.73	**908.89**	−0.64%
11	399	1000	**1139.51**	1146.57	**1139.51**	−0.62%
12	483	1000	**1384.29**	1386.48	**1384.29**	−0.16%
13	252	1000	**1030.42**	1047.57	**1030.42**	−1.64%
14	320	1000	**1324.96**	1340.16	**1324.96**	−1.13%
15	396	1000	**1668.39**	1700.28	**1668.39**	−1.88%
16	480	1000	**2053.47**	2097.47	**2053.47**	−2.10%
17	240	200	**840.53**	867.03	**840.53**	−3.06%
18	300	200	**1097.51**	1104.86	**1097.51**	−0.67%
19	360	200	**1522.83**	**1522.83**	1545.53	1.49%
20	420	200	**2019.55**	**2019.55**	2042.90	1.16%
#best solutions				2	18	

Table 6
Results for the adapted problem instances by Golden et al. (1998) with $\rho = 25\%$. Comparison between the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper.

Index	n+1	Q	Best known	Expósito-Izquierdo et al. (2016)	Two-level VNS	Gap
1	240	550	**6051.04**	6135.26	**6051.04**	−1.37%
2	320	700	**9725.90**	10005.59	**9725.90**	−2.80%
3	400	900	**13692.60**	14083.28	**13692.60**	−2.77%
4	480	1000	**16977.90**	17359.95	**16977.90**	−2.20%
5	200	900	**9340.70**	9701.89	**9340.70**	−3.72%
6	280	900	**10840.70**	11261.49	**10840.70**	−3.74%
7	360	900	**12348.10**	12720.79	**12348.10**	−2.93%
8	440	900	**14100.80**	14307.64	**14100.80**	−1.45%
9	255	1000	**717.63**	723.49	**717.63**	−0.81%
10	323	1000	**908.26**	915.09	**908.26**	−0.75%
11	399	1000	**1131.84**	1140.36	**1131.84**	−0.75%
12	483	1000	**1387.67**	1395.67	**1387.67**	−0.57%
13	252	1000	**1034.30**	1054.64	**1034.30**	−1.93%
14	320	1000	**1317.05**	1341.39	**1317.05**	−1.81%
15	396	1000	**1667.08**	1697.88	**1667.08**	−1.81%
16	480	1000	**2048.08**	2105.01	**2048.08**	−2.70%
17	240	200	**795.33**	808.24	**795.33**	−1.60%
18	300	200	**1122.73**	1138.45	**1122.73**	−1.38%
19	360	200	**1538.20**	1549.89	**1538.20**	−0.75%
20	420	200	**2036.19**	**2036.19**	2038.27	0.10%
#best solutions				1	19	

times can be considered an important drawback for some applications.

To allow the comparison with the results reported by Expósito-Izquierdo et al. (2016), we dedicate a maximum calculation time of 10 seconds to our two-level VNS. On average, this corresponds to a 98% reduction in calculation time compared to Vidal et al. (2015). The two-level VNS is able to find the optimal solution for only 8 out of 220 instances. An optimality gap of around 1% is obtained on average, which is equivalent to a reduction of the optimality gap by 63% compared to Expósito-Izquierdo et al. (2016).

5. The CluVRP with weak cluster constraints

5.1. Motivation

As described above, the CluVRP with strong cluster constraints requires that all customers that belong to the same cluster should be served consecutively by the same vehicle. This requirement can be relaxed in some real life applications. In parcel delivery, e.g., the customers are often clustered in *zones* (clusters) in order to facilitate the sorting process. These zones are assigned to the available vehicles, obtaining a *tactical plan* in which each vehicle is allowed to serve multiple zones during one trip. (Janssens et al., 2015) However, from the moment that the vehicle leaves the depots there is no need to visit the individual customers according to their original zone. It might be profitable for the driver to leave a current zone, serve customers belonging to another zone and return to the initial zone afterwards. This can depend on the layout of the instance and the individual customer locations, but also on real time traffic information or additional constraints such as time windows. This gives rise to another variant of the CluVRP, which we define as the CluVRP*with weak cluster constraints*. Similar to the CluVRP with strong cluster constraints, we impose that clusters are assigned to vehicles and therefore that all customers that belong to a certain cluster are all served by the same vehicle. However, we allow a vehicle to leave and re-enter a cluster multiple times during its trip.

Table 7
Results for the adapted problem instances by Golden et al. (1998) with $\rho = 50\%$. Comparison between the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper.

Index	n+1	Q	Best known	Expósito-Izquierdo et al. (2016)	Two-level VNS	Gap
1	240	550	**6551.04**	6719.17	**6551.04**	−2.50%
2	320	700	**9787.09**	9904.40	**9787.09**	−1.18%
3	400	900	**13287.00**	13303.31	**13287.00**	−0.12%
4	480	1000	**17569.90**	17935.58	**17569.90**	−2.04%
5	200	900	**8597.31**	8790.44	**8597.31**	−2.20%
6	280	900	**10550.80**	10714.34	**10550.80**	−1.53%
7	360	900	**12673.70**	12862.90	**12673.70**	−1.47%
8	440	900	**13766.30**	13924.79	**13766.30**	−1.14%
9	255	1000	**698.04**	703.07	**698.04**	−0.72%
10	323	1000	**890.87**	898.19	**890.87**	−0.81%
11	399	1000	**1107.88**	1112.35	**1107.88**	−0.40%
12	483	1000	**1317.47**	1319.98	**1317.47**	−0.19%
13	252	1000	**1053.47**	1080.84	**1053.47**	−2.53%
14	320	1000	**1342.70**	1363.99	**1342.70**	−1.56%
15	396	1000	**1657.22**	1685.61	**1657.22**	−1.68%
16	480	1000	**2003.10**	2030.60	**2003.10**	−1.35%
17	240	200	**881.66**	910.73	**881.66**	−3.19%
18	300	200	**1200.98**	1217.71	**1200.98**	−1.37%
19	360	200	**1612.33**	1631.21	**1612.33**	−1.16%
20	420	200	**2278.64**	2325.47	**2278.64**	−2.01%
#best solutions				0	20	

Table 8
Results for the adapted problem instances by Golden et al. (1998) with $\rho = 75\%$. Comparison between the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper.

Index	n+1	Q	Best known	Expósito-Izquierdo et al. (2016)	Two-level VNS	Gap
1	240	550	**6736.15**	**6736.15**	**6736.15**	0.00%
2	320	700	**10204.30**	**10204.30**	10223.20	0.19%
3	400	900	**13575.70**	**13575.70**	13635.20	0.44%
4	480	1000	**17077.59**	**17077.59**	17194.20	0.68%
5	200	900	**8664.94**	**8664.94**	8666.59	0.02%
6	280	900	**11452.01**	**11452.01**	11520.30	0.60%
7	360	900	**12901.41**	**12901.41**	12950.00	0.38%
8	440	900	**13926.40**	13943.65	**13926.40**	−0.12%
9	255	1000	**773.39**	**773.39**	**773.39**	0.00%
10	323	1000	**1000.51**	**1000.51**	1001.28	0.08%
11	399	1000	**1223.66**	**1223.66**	1226.91	0.27%
12	483	1000	**1475.68**	**1475.68**	1478.86	0.22%
13	252	1000	**1183.12**	**1183.12**	**1183.12**	0.00%
14	320	1000	**1520.55**	1523.44	**1520.55**	−0.19%
15	396	1000	**1825.29**	1829.32	**1825.29**	−0.22%
16	480	1000	**2265.54**	**2265.54**	2265.77	0.01%
17	240	200	**1001.02**	**1001.02**	**1001.02**	0.00%
18	300	200	**1392.15**	1396.27	**1392.15**	−0.29%
19	360	200	**1951.77**	1977.40	**1951.77**	−1.30%
20	420	200	**2540.22**	**2540.22**	2540.39	0.01%
#best solutions				14	9	

5.2. Mathematical model

The mathematical model, introduced in Section 2 is adapted below to comply with the weak cluster constraints. Constraints (7) are relaxed and replace by constraints (16) as each cluster can now be visited multiple times. Furthermore, a additional set of equations is added to the model to ensure that all customers that belong to the same cluster are visited by the same vehicle (see constraints (17)).

$$\min \sum_{(i,j)\in E} \sum_{k\in K} d_{ij} x_{ijk} \tag{10}$$

Subject to

$$\sum_{k\in K} y_{ik} = 1 \qquad \forall i \in V\backslash V_0 \tag{11}$$

$$\sum_{k\in K} y_{0k} = |K| \tag{12}$$

$$\sum_{j\in V\backslash V_0} x_{ijk} = \sum_{j\in V\backslash V_0} x_{jik} = y_{ik} \qquad \forall k \in K, \forall i \in V \tag{13}$$

$$\sum_{i\in V} q_i y_{ik} \le Q \qquad \forall k \in K \tag{14}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijk} \le y_{hk} \qquad \forall Z \subseteq V\backslash V_0, \forall h \in Z, \forall k \in K \tag{15}$$

$$\sum_{(i,j)\in \delta^+(C_r)} x_{ijk} = \sum_{(i,j)\in \delta^-(C_r)} x_{ijk} \ge 1 \qquad \forall r \in R \tag{16}$$

Table 9
Results for the adapted problem instances by Golden et al. (1998) with $\rho = 100\%$. Comparison between the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper.

Index	n+1	Q	Best known	Expósito-Izquierdo et al. (2016)	Two-level VNS	Gap
1	240	550	**6293.04**	**6293.04**	6297.05	0.06%
2	320	700	**9879.59**	**9879.59**	9917.68	0.39%
3	400	900	**12361.09**	**12361.09**	12422.10	0.49%
4	480	1000	**16130.39**	**16130.39**	16276.70	0.91%
5	200	900	**8394.11**	**8394.11**	8399.31	0.06%
6	280	900	**10777.33**	**10777.33**	10802.70	0.24%
7	360	900	**11346.11**	**11346.11**	11411.60	0.58%
8	440	900	**13188.94**	**13188.94**	13251.30	0.47%
9	255	1000	**705.19**	**705.19**	**705.19**	0.00%
10	323	1000	**837.52**	**837.52**	838.55	0.12%
11	399	1000	**1054.13**	**1054.13**	1056.71	0.24%
12	483	1000	**1297.31**	**1297.31**	1300.02	0.21%
13	252	1000	**996.36**	**996.36**	**996.36**	0.00%
14	320	1000	**1223.09**	**1223.09**	1223.41	0.03%
15	396	1000	**1531.29**	**1531.29**	1532.82	0.10%
16	480	1000	**1874.69**	**1874.69**	1875.04	0.02%
17	240	200	**844.27**	**844.27**	**844.27**	0.00%
18	300	200	**1212.97**	**1212.97**	1213.13	0.01%
19	360	200	**1667.45**	**1667.45**	1667.51	0.00%
20	420	200	**2128.60**	**2128.60**	2128.77	0.01%
#best solutions				20	3	

Table 10
Comparison between the UHGS$_p$ algorithms from Vidal et al. (2015), the two-level approach from Expósito-Izquierdo et al. (2016) and the two-level VNS proposed in this paper over the problem instances proposed by Battarra et al. (2014).

n+1	Vidal et al. (2015)		Expósito-Izquierdo et al. (2016)		Two-level VNS		
	Gap	t(s)	Gap	t(s)	Gap	t(s)	Gap reduction
200	0.00%	2866.56	4.61%	10.0	0.07%	10.0	−98.56%
241	0.00%	174.90	2.39%	10.0	0.44%	10.0	−81.66%
252	0.01%	164.69	0.50%	10.0	0.53%	10.0	5.97%
255	0.02%	135.45	3.69%	10.0	1.33%	10.0	−63.93%
280	0.00%	3848.31	2.94%	10.0	0.71%	10.0	−76.00%
300	0.00%	191.26	1.04%	10.0	0.93%	10.0	−11.05%
320	0.02%	198.09	1.26%	10.0	0.85%	10.0	−32.75%
323	0.08%	175.74	4.94%	10.0	0.93%	10.0	−81.22%
360	0.00%	1248.29	2.87%	10.0	1.02%	10.0	−64.52%
396	0.05%	279.15	1.54%	10.0	1.37%	10.0	−10.89%
399	0.06%	198.00	4.96%	10.0	2.15%	10.0	−56.58%
400	0.01%	1384.18	2.56%	10.0	1.26%	10.0	−50.61%
420	0.00%	351.74	2.60%	10.0	1.11%	10.0	−57.22%
440	0.02%	1017.64	3.67%	10.0	1.32%	10.0	−64.02%
480	0.01%	1427.23	3.42%	10.0	1.49%	10.0	−56.38%
483	0.07%	389.16	4.93%	10.0	2.23%	10.0	−54.75%
Average	0.02%	878.15	3.00%	10.0	1.11%	10.0	−62.99%

$$y_{ik} = y_{jk} \qquad \forall i, j \in C_r, \forall k \in K \tag{17}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall (i, j) \in E, \forall k \in K \tag{18}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V, \forall k \in K \tag{19}$$

5.3. Experiments

Our two-level VNS approach is slightly altered to solve the CluVRP with weak cluster constraints. More specifically, the intra cluster neighborhoods of the VNS at the customer level are expanded to inter cluster operators, allowing the customers to be moved to any position in the trip.

Again, the GVRPθ3 and Golden instances as proposed by Battarra et al. (2014) are solved. Table 11 bundles the differences in total distance between the CluVRP with strong and weak cluster constraints. These relative differences are obtained by comparing the results of our algorithm for weak cluster constraints to the results discussed in Section 4.3.

As in this section the problem becomes less constrained, the objective value, defined as the total distance travelled by all vehicles, is lower for all instance classes compared to the CluVRP with hard cluster constraints. The reduction in objective value lies between 4 and 7% on average for the Golden instances, but can go up to more than 12% for some of the instances. As the choice between strong and weak cluster constraints might result in significant differences in total cost, this decision should be taken with care in real life scenarios.

6. Conclusion

In this paper, a fast two-level VNS heuristic was presented to solve the Clustered Vehicle Routing Problem. This metaheuristic

Table 11
Results for the GVRPθ3 and Golden instances as proposed by Battarra et al. (2014) with weak cluster constraints. Reported values are the averaged over all instances in the set, compared to the obtained results with strong cluster constraints.

GVRPθ3				Golden instances			
Instance set	t(s)	Avg. best difference	Avg. difference	n+1	t(s)	Avg. best difference	Avg. difference
A	0.28	−2.66%	−2.52%	200	10.00	−8.97%	−8.09%
B	0.06	−1.18%	−1.18%	241	10.00	−5.69%	−4.92%
P	0.52	−4.64%	−4.58%	252	10.00	−3.62%	−3.04%
M+G	13.46	−3.19%	−2.73%	255	10.00	−5.02%	−3.88%
				280	10.00	−6.57%	−5.56%
				300	10.00	−3.81%	−3.09%
				320	10.00	−3.46%	−2.88%
				323	10.00	−6.10%	−4.98%
				360	10.00	−5.23%	−4.48%
				396	10.00	−3.13%	−2.41%
				399	10.00	−4.67%	−3.48%
				400	10.00	−3.81%	−3.20%
				420	10.00	−3.49%	−2.71%
				440	10.00	−4.25%	−3.37%
				480	10.00	−3.87%	−3.15%
				483	10.00	−2.96%	−1.63%
average	1.12	−2.86%	−2.80%			−4.65%	−3.82%

solution approach solves the CLuVRP without assuming Euclidean distances or converting the problem to a CVRP by using a big-M approach. By integrating a cluster level and a customer level local search phase, the specific clustered structure of the problem was exploited to reduce complexity, and high quality solutions could be obtained in very short calculation times.

Our algorithm was tested on benchmark instances from the literature with different sizes and diverse complexities. Many of the small and medium-sized problems were solved to optimality in very short computing times. Even if the optimal solution was not found, an average optimality gap of 0.03% was obtained. For the large-size instances, which are adaptations of the Golden benchmark instances as proposed by Battarra et al. (2014) and Expósito-Izquierdo et al. (2016), only a limited number of optimal solutions were found. However, with an average optimality gap around 1% high quality and competitive solutions were obtained by our two-level VNS approach in very small computation times. We therefore believe that our solution approach has potential for integration in solution approaches that rely on the CLuVRP as a subproblem as this might require the method to be executed multiple times. For example, a courier company that wants to determine the boundaries of its zones (cluster) in the distribution area might want to solve the CLuVRP for every possible configuration to select the best option. In this context, a fast method is preferred while the remaining optimality gap is a minor issue. By allowing larger calculation times, the optimality gap is likely to reduce further. However, we leave this trade-off to the decision maker.

Furthermore, we have introduced a new type of CLuVRP in this paper. Next to the traditional CLuVRP with strong cluster constraints, in which it is not allowed to leave a cluster before having served all customers within it, we have proposed a CLuVRP variant with weak cluster constraints. Here, all customers belonging to the same cluster should be served by the same vehicle but customers in the different clusters assigned to a vehicle can be visited in any order. Our simulation experiments show that the total distance travelled might decreases by 4.65% on average for the large-size instances when going from strong to weak cluster constraints. For some instances, a cost reduction of more than 10% could be obtained. These values put an estimate on the profit of allowing the approach with weak cluster constraints as opposed to strong cluster constraints, e.g., in the context of the operations of a courier company.

Although all algorithms are tested and compared on multiple sets of benchmark instances, a gap remains between the size of these instances and the complexity of the logistic problems faced by the industry today. For further research, we therefore acknowledge the need for additional very large-size instances that are able to better represent the daily planning problems faced by, e.g., parcel delivery and courier companies.

Acknowledgements

This research is supported bythe 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.

Appendix A. Detailed results

Table 12
Detailed results for the GVRPθ3 instances A, B.

Instance						Strong cluster constraint				Weak cluster constraints	
Set	n	k	c	v	opt.	Best	Avg. best GAP	Avg. GAP	Avg. CPU(s)	Avg. best GAP	Avg. GAP
A	32	5	11	2	522	**522**	**0.00%**	**0.00%**	0.00	−1.34%	−1.34%
A	33	5	11	2	472	**472**	**0.00%**	**0.00%**	0.00	−2.33%	−2.33%
A	33	6	11	2	562	**562**	**0.00%**	**0.00%**	0.00	−1.42%	−1.42%
A	34	5	12	2	547	**547**	**0.00%**	**0.00%**	0.00	−1.65%	−1.65%
A	36	5	12	2	588	589	0.17%	0.17%	0.00	−7.81%	−7.81%

(*continued on next page*)

Table 12 (*continued*)

Instance						Strong cluster constraint				Weak cluster constraints	
Set	n	k	c	v	opt.	Best	Avg. best GAP	Avg. GAP	Avg. CPU(s)	Avg. best GAP	Avg. GAP
A	37	5	13	2	569	**569**	**0.00%**	**0.00%**	0.00	−2.46%	−2.46%
A	37	6	13	2	615	**615**	**0.00%**	**0.00%**	0.02	−1.63%	−1.56%
A	38	5	13	2	507	**507**	**0.00%**	**0.00%**	0.00	0.00%	0.00%
A	39	5	13	2	610	**610**	**0.00%**	**0.00%**	0.02	−3.61%	−3.41%
A	39	6	13	2	613	**613**	**0.00%**	**0.00%**	0.00	−1.63%	−1.63%
A	44	6	15	2	714	**714**	**0.00%**	**0.00%**	0.04	−3.22%	−3.22%
A	45	6	15	3	712	**712**	**0.00%**	**0.00%**	0.00	−8.43%	−8.43%
A	45	7	15	3	664	**664**	**0.00%**	**0.00%**	0.00	−0.45%	−0.45%
A	46	7	16	3	664	**664**	**0.00%**	**0.00%**	0.00	−3.31%	−3.31%
A	48	7	16	3	683	**683**	**0.00%**	**0.00%**	0.00	−0.44%	−0.44%
A	53	7	18	3	651	**651**	**0.00%**	**0.00%**	0.00	−3.69%	−3.69%
A	54	7	18	3	724	**724**	**0.00%**	**0.00%**	0.01	−3.45%	−3.45%
A	55	9	19	3	653	**653**	**0.00%**	**0.00%**	0.00	−1.23%	−1.23%
A	60	9	20	3	787	795	1.02%	1.02%	0.02	−4.15%	−4.04%
A	61	9	21	4	682	**682**	**0.00%**	**0.00%**	0.00	−1.61%	−1.11%
A	62	8	21	3	778	**778**	**0.00%**	**0.00%**	0.01	−0.90%	−0.90%
A	63	10	21	4	801	**801**	**0.00%**	**0.00%**	0.02	−2.75%	−2.65%
A	63	9	21	3	865	**865**	**0.00%**	**0.00%**	0.45	−3.24%	−3.21%
A	64	9	22	3	773	**773**	**0.00%**	**0.00%**	0.09	−0.78%	−0.78%
A	65	9	22	3	725	**725**	**0.00%**	0.01%	0.15	−4.41%	−4.41%
A	69	9	23	3	814	819	0.61%	0.66%	0.38	−3.05%	−2.37%
A	80	10	27	4	972	**972**	**0.00%**	0.10%	0.27	−2.88%	−2.78%
B	31	5	11	2	375	**375**	**0.00%**	**0.00%**	0.00	0.00%	0.00%
B	34	5	12	2	416	**416**	**0.00%**	**0.00%**	0.00	−0.24%	−0.24%
B	35	5	12	2	562	**562**	**0.00%**	0.16%	0.03	−0.89%	−0.89%
B	38	6	13	2	431	**431**	**0.00%**	**0.00%**	0.06	−0.93%	−0.93%
B	39	5	13	2	321	**321**	**0.00%**	**0.00%**	0.00	−1.25%	−1.25%
B	41	6	14	2	476	**476**	**0.00%**	**0.00%**	0.00	−1.47%	−1.47%
B	43	6	15	2	415	**415**	**0.00%**	**0.00%**	0.00	−2.41%	−2.41%
B	44	7	15	3	447	**447**	**0.00%**	**0.00%**	0.00	−0.89%	−0.89%
B	45	5	15	2	506	508	0.40%	0.41%	0.02	−3.74%	−3.74%
B	45	6	15	2	391	**391**	**0.00%**	**0.00%**	0.02	−1.28%	−1.28%
B	50	7	17	3	467	**467**	**0.00%**	**0.00%**	0.00	−0.64%	−0.64%
B	50	8	17	3	666	**666**	**0.00%**	**0.00%**	0.02	−0.75%	−0.75%
B	51	7	17	3	585	**585**	**0.00%**	**0.00%**	0.00	−1.20%	−1.20%
B	52	7	18	3	427	**427**	**0.00%**	**0.00%**	0.00	0.00%	0.00%
B	56	7	19	3	433	434	0.23%	0.23%	0.01	−3.23%	−3.23%
B	57	7	19	3	634	**634**	**0.00%**	**0.00%**	0.01	−1.89%	−1.89%
B	57	9	19	3	753	**753**	**0.00%**	**0.00%**	0.01	−0.93%	−0.93%
B	63	10	21	3	685	**685**	**0.00%**	**0.00%**	0.06	0.00%	0.00%
B	64	9	22	4	526	**526**	**0.00%**	**0.00%**	0.00	−0.38%	−0.38%
B	66	9	22	3	687	**687**	**0.00%**	**0.00%**	0.26	−0.58%	−0.58%
B	67	10	23	4	626	**626**	**0.00%**	**0.00%**	0.04	−1.12%	−1.12%
B	68	9	23	3	588	**588**	**0.00%**	0.03%	0.29	−1.02%	−1.02%
B	78	10	26	4	721	**721**	**0.00%**	**0.00%**	0.02	−2.36%	−2.36%

Table 13
Detailed results for the GVRPθ3 instances G, M, P.

Instance						Strong cluster constraint				Weak cluster constraints	
Set	n	k	c	v	opt.	Best	Avg. best GAP	Avg. GAP	Avg. CPU(s)	Avg. Best GAP	Avg. GAP
G	262	25	88	9	–	**3310**			15.95	−3.44%	−2.71%
M	101	10	34	4	607	**607**	**0.00%**	**0.00%**	0.03	−1.48%	−1.48%
M	121	7	41	3	691	**691**	**0.00%**	0.35%	4.25	−1.45%	−0.94%
M	151	12	51	4	804	805	0.12%	0.19%	3.20	−5.71%	−5.11%
M	200	16	67	6	914 (UB)	**909**	**−0.55%**	**−0.34%**	6.45	−3.85%	−3.40%
P	101	4	34	2	679	**679**	**0.00%**	**0.00%**	0.20	−4.42%	−4.32%
P	16	8	6	4	253	**253**	**0.00%**	**0.00%**	0.00	−0.79%	−0.79%
P	19	2	7	1	186	**186**	**0.00%**	**0.00%**	0.00	−8.60%	−8.60%
P	20	2	7	1	200	**200**	**0.00%**	**0.00%**	0.00	−11.50%	−11.50%
P	21	2	7	1	190	**190**	**0.00%**	**0.00%**	0.00	−5.79%	−5.79%
P	22	2	8	1	202	**202**	**0.00%**	**0.00%**	0.00	−9.41%	−9.41%
P	22	8	8	4	365	**365**	**0.00%**	**0.00%**	0.00	0.00%	0.00%
P	23	8	8	3	279	**279**	**0.00%**	**0.00%**	0.00	−3.23%	−3.23%
P	40	5	14	2	396	**396**	**0.00%**	**0.00%**	0.00	−3.79%	−3.79%
P	45	5	15	2	440	**440**	**0.00%**	**0.00%**	0.00	−4.09%	−4.09%
P	50	10	17	4	491	**491**	**0.00%**	**0.00%**	0.02	−4.07%	−4.07%
P	50	7	17	3	447	**447**	**0.00%**	**0.00%**	0.02	−3.80%	−3.80%
P	50	8	17	3	460	**460**	**0.00%**	**0.00%**	0.01	−4.13%	−4.13%
P	51	10	17	4	537	**537**	**0.00%**	0.02%	0.03	−8.19%	−8.19%
P	55	10	19	4	500	**500**	**0.00%**	**0.00%**	0.03	−3.80%	−3.75%
P	55	15	19	6	595	**595**	**0.00%**	**0.00%**	0.01	−3.87%	−3.87%
P	55	7	19	3	462	**462**	**0.00%**	**0.00%**	0.00	−1.30%	−1.30%
P	55	8	19	3	471	**471**	**0.00%**	**0.00%**	0.02	−3.18%	−3.18%
P	60	10	20	4	552	**552**	**0.00%**	0.03%	0.19	−3.08%	−2.96%
P	60	15	20	5	611	**611**	**0.00%**	0.18%	0.29	−3.27%	−3.27%
P	65	10	22	4	619	**619**	**0.00%**	**0.00%**	0.01	−5.98%	−5.98%
P	70	10	24	4	643	644	0.16%	0.16%	0.01	−6.52%	−6.12%
P	76	4	26	2	581	**581**	**0.00%**	**0.00%**	0.41	−4.13%	−3.81%
P	76	5	26	2	581	**581**	**0.00%**	**0.00%**	0.31	−4.30%	−4.00%

Table 14
Detailed results for the Golden instances 1–5, as provided by Battarra et al. (2014).

Instance				Strong cluster constraints				Weak cluster constraints	
Set	n	N	opt.	Best	Avg. best GAP(%)	Avg. GAP(%)	Avg. CPU(s)	Best diff.	Avg. Diff.
Golden-1	241	17	4831	4862	0.64%	1.19%	5.33	−3.39%	−2.38%
Golden-1	241	18	4847	4864	0.35%	0.98%	4.68	−2.80%	−1.97%
Golden-1	241	19	4872	4889	0.35%	1.24%	4.05	−2.92%	−2.41%
Golden-1	241	21	4889	4914	0.51%	1.13%	5.65	−2.81%	−2.43%
Golden-1	241	22	4908	4950	0.86%	1.20%	4.07	−4.36%	−3.21%
Golden-1	241	25	4899	4917	0.37%	0.80%	3.72	−3.72%	−3.21%
Golden-1	241	27	4934	4952	0.36%	0.67%	4.14	−4.81%	−3.70%
Golden-1	241	31	5050	5053	0.06%	0.19%	5.21	−5.68%	−4.99%
Golden-1	241	35	5102	5116	0.27%	0.54%	4.75	−7.74%	−6.81%
Golden-1	241	41	5097	5113	0.31%	0.74%	4.85	−7.88%	−6.76%
Golden-1	241	49	5000	5039	0.78%	1.35%	4.24	−7.32%	−5.43%
Golden-2	321	22	7716	7785	0.89%	1.24%	6.19	−2.84%	−2.43%
Golden-2	321	23	7693	7768	0.97%	1.36%	5.29	−2.79%	−2.43%
Golden-2	321	25	7668	7728	0.78%	1.15%	5.79	−3.35%	−2.44%
Golden-2	321	27	7638	7705	0.88%	1.23%	4.50	−2.92%	−2.39%
Golden-2	321	30	7617	7689	0.95%	1.51%	5.51	−2.65%	−2.26%
Golden-2	321	33	7640	7705	0.85%	1.24%	4.84	−3.04%	−2.70%
Golden-2	321	36	7643	7699	0.73%	1.14%	3.62	−3.27%	−2.79%
Golden-2	321	41	7738	7781	0.56%	1.16%	5.08	−4.25%	−3.46%
Golden-2	321	46	7861	7926	0.83%	1.30%	4.43	−5.41%	−4.86%
Golden-2	321	54	7920	7989	0.87%	1.28%	3.84	−6.28%	−5.62%
Golden-2	321	65	7892	7997	1.33%	1.71%	5.45	−6.50%	−5.67%
Golden-3	401	27	10540	10662	1.16%	1.80%	4.81	−2.56%	−2.17%
Golden-3	401	29	10504	10627	1.17%	1.50%	5.33	−3.72%	−2.92%
Golden-3	401	31	10486	10616	1.24%	1.48%	4.68	−3.23%	−2.79%
Golden-3	401	34	10465	10602	1.31%	1.64%	5.18	−2.92%	−2.60%
Golden-3	401	37	10482	10605	1.17%	1.66%	4.25	−3.30%	−2.75%
Golden-3	401	41	10501	10606	1.00%	1.51%	4.87	−3.12%	−2.70%
Golden-3	401	45	10485	10649	1.56%	1.78%	4.56	−3.73%	−2.98%
Golden-3	401	51	10583	10722	1.31%	2.00%	4.46	−4.01%	−3.43%
Golden-3	401	58	10776	10905	1.20%	1.71%	4.75	−5.09%	−4.48%
Golden-3	401	67	10797	10953	1.44%	1.79%	4.51	−6.06%	−4.96%
Golden-3	401	81	10614	10756	1.34%	1.91%	4.96	−4.14%	−3.44%
Golden-4	481	33	13598	13805	1.52%	1.99%	5.07	−4.97%	−3.95%
Golden-4	481	35	13643	13795	1.11%	1.89%	4.35	−4.28%	−3.71%
Golden-4	481	37	13520	13722	1.49%	1.83%	5.32	−4.59%	−3.80%
Golden-4	481	41	13460	13618	1.17%	1.91%	5.09	−4.46%	−3.52%
Golden-4	481	44	13568	13756	1.39%	1.70%	5.25	−4.66%	−3.97%
Golden-4	481	49	13758	13968	1.53%	2.10%	4.93	−5.98%	−5.19%
Golden-4	481	54	13760	13985	1.64%	2.27%	5.62	−6.16%	−5.02%
Golden-4	481	61	13791	14045	1.84%	2.25%	5.51	−6.09%	−4.98%
Golden-4	481	69	13966	14143	1.27%	2.00%	4.31	−6.00%	−5.35%
Golden-4	481	81	13975	14167	1.37%	2.13%	5.32	−6.58%	−5.75%
Golden-4	481	97	13775	13973	1.44%	2.22%	3.50	−4.46%	−3.96%
Golden-5	201	14	7622	7652	0.39%	0.74%	4.62	−7.08%	−6.39%
Golden-5	201	15	7424	7429	0.07%	0.67%	4.52	−8.49%	−7.55%
Golden-5	201	16	7491	**7491**	**0.00%**	0.30%	4.89	−8.92%	−8.00%
Golden-5	201	17	7434	**7434**	**0.00%**	0.44%	4.81	−7.61%	−7.04%
Golden-5	201	19	7576	**7576**	**0.00%**	0.08%	4.14	−7.23%	−6.69%
Golden-5	201	21	7596	**7596**	**0.00%**	0.03%	4.59	−9.00%	−8.39%
Golden-5	201	23	7643	**7643**	**0.00%**	0.24%	2.38	−11.33%	−10.18%
Golden-5	201	26	7560	7566	0.08%	0.21%	3.66	−11.12%	−10.20%
Golden-5	201	29	7410	**7410**	**0.00%**	0.04%	4.91	−8.99%	−8.11%
Golden-5	201	34	7429	7433	0.05%	0.17%	5.28	−9.66%	−8.32%
Golden-5	201	41	7241	7251	0.14%	0.25%	5.39	−9.20%	−8.11%

Table 15
Detailed results for the Golden instances 6–10, as provided by Battarra et al. (2014).

Instance				Strong cluster constraints				Weak cluster constraints	
Set	n	N	opt.	Best	Avg. best GAP(%)	Avg. GAP(%)	Avg. CPU(s)	Best diff.	Avg. Diff.
Golden-6	281	19	8624	8685	0.71%	0.94%	4.72	−5.62%	−4.91%
Golden-6	281	21	8628	8661	0.38%	0.73%	5.04	−5.83%	−4.38%
Golden-6	281	22	8646	8715	0.80%	1.26%	6.02	−5.96%	−4.67%
Golden-6	281	24	8853	8905	0.59%	1.04%	3.59	−5.44%	−4.63%
Golden-6	281	26	8910	8978	0.76%	1.31%	4.16	−6.32%	−5.52%
Golden-6	281	29	8936	9025	1.00%	1.46%	4.69	−6.98%	−6.08%
Golden-6	281	32	8891	8974	0.93%	1.44%	3.93	−7.62%	−5.90%
Golden-6	281	36	8969	9011	0.47%	0.82%	4.89	−6.97%	−6.36%
Golden-6	281	41	9028	9067	0.43%	0.80%	5.37	−7.30%	−6.42%
Golden-6	281	47	8923	8996	0.82%	1.37%	2.99	−7.19%	−6.10%
Golden-6	281	57	9028	9107	0.88%	1.32%	5.36	−7.09%	−6.23%
Golden-7	361	25	9904	10021	1.18%	1.68%	4.32	−4.42%	−3.72%
Golden-7	361	26	9888	10023	1.37%	1.76%	5.61	−4.42%	−3.71%
Golden-7	361	28	9917	10056	1.40%	1.84%	5.39	−4.72%	−4.02%
Golden-7	361	31	10021	10131	1.10%	1.53%	4.67	−3.99%	−3.58%
Golden-7	361	33	10029	10161	1.32%	1.57%	5.07	−4.68%	−4.02%
Golden-7	361	37	10131	10176	0.44%	1.14%	4.67	−4.93%	−3.75%
Golden-7	361	41	10052	10119	0.67%	1.27%	4.94	−4.50%	−3.61%
Golden-7	361	46	10080	10197	1.16%	1.77%	5.75	−5.44%	−4.24%
Golden-7	361	52	10095	10201	1.05%	1.61%	4.40	−4.97%	−4.10%
Golden-7	361	61	10096	10189	0.92%	1.74%	5.74	−4.67%	−4.18%
Golden-7	361	73	10014	10095	0.81%	1.52%	4.70	−4.88%	−3.57%
Golden-8	441	30	10866	11002	1.25%	1.58%	4.86	−3.19%	−2.45%
Golden-8	441	32	10831	10943	1.03%	1.73%	5.82	−2.77%	−2.07%
Golden-8	441	34	10847	10963	1.07%	1.68%	5.33	−2.56%	−2.04%
Golden-8	441	37	10859	11010	1.39%	1.95%	5.06	−3.18%	−2.59%
Golden-8	441	41	10934	11088	1.41%	1.79%	5.94	−3.57%	−3.02%
Golden-8	441	45	10960	11103	1.30%	1.60%	4.48	−3.93%	−3.00%
Golden-8	441	49	11042	11177	1.22%	1.61%	4.81	−3.98%	−3.34%
Golden-8	441	56	11194	11350	1.39%	1.86%	4.86	−5.50%	−4.48%
Golden-8	441	63	11252	11412	1.42%	1.76%	4.64	−5.83%	−4.51%
Golden-8	441	74	11321	11462	1.25%	2.09%	4.50	−6.17%	−4.84%
Golden-8	441	89	11209	11409	1.78%	2.45%	5.08	−6.09%	−4.74%
Golden-9	256	18	300	304	1.33%	1.77%	4.62	−5.26%	−4.18%
Golden-9	256	19	299	304	1.67%	1.96%	3.67	−5.59%	−3.98%
Golden-9	256	20	296	298	0.68%	1.59%	3.09	−3.69%	−2.67%
Golden-9	256	22	290	295	1.72%	2.45%	3.98	−3.39%	−2.85%
Golden-9	256	24	290	295	1.72%	2.48%	3.84	−3.73%	−2.78%
Golden-9	256	26	288	293	1.74%	2.20%	5.29	−3.41%	−2.92%
Golden-9	256	29	292	297	1.71%	2.40%	3.21	−5.39%	−4.09%
Golden-9	256	32	297	300	1.01%	1.58%	4.38	−6.67%	−5.02%
Golden-9	256	37	294	296	0.68%	1.55%	3.31	−5.41%	−3.89%
Golden-9	256	43	295	300	1.69%	2.36%	2.97	−6.33%	−5.27%
Golden-9	256	52	296	298	0.68%	2.50%	4.30	−6.38%	−5.05%
Golden-10	324	22	367	369	0.54%	1.20%	2.41	−3.25%	−1.88%
Golden-10	324	24	361	**361**	**0.00%**	0.43%	5.03	−1.94%	−0.83%
Golden-10	324	25	359	360	0.28%	0.88%	3.75	−1.11%	−0.58%
Golden-10	324	27	361	363	0.55%	1.65%	3.53	−1.65%	−1.17%
Golden-10	324	30	367	371	1.09%	1.58%	4.31	−3.77%	−2.87%
Golden-10	324	33	373	378	1.34%	2.25%	2.51	−5.82%	−4.80%
Golden-10	324	36	385	391	1.56%	2.04%	4.22	−8.70%	−7.43%
Golden-10	324	41	400	403	0.75%	1.44%	4.72	−9.93%	−9.09%
Golden-10	324	47	398	402	1.01%	1.58%	4.65	−10.45%	−9.38%
Golden-10	324	54	393	397	1.02%	1.95%	3.92	−10.08%	−8.66%
Golden-10	324	65	387	395	2.07%	2.70%	3.64	−10.38%	−8.09%

Table 16
Detailed results for the Golden instances 11–15, as provided by Battarra et al. (2014).

Instance				Strong cluster constraints				Weak cluster constraints	
Set	n	N	opt.	Best	Avg. best GAP(%)	Avg. GAP(%)	Avg. CPU(s)	Best diff.	Avg. Diff.
Golden-11	400	27	457	464	1.53%	1.76%	3.58	−3.66%	−2.73%
Golden-11	400	29	455	463	1.76%	2.38%	3.58	−3.46%	−2.71%
Golden-11	400	31	455	464	1.98%	2.59%	5.29	−3.66%	−2.66%
Golden-11	400	34	455	462	1.54%	2.21%	4.23	−4.55%	−3.19%
Golden-11	400	37	459	469	2.18%	2.68%	4.65	−5.33%	−4.08%
Golden-11	400	40	461	467	1.30%	2.25%	3.87	−4.93%	−3.88%
Golden-11	400	45	462	470	1.73%	2.54%	4.69	−5.96%	−4.66%
Golden-11	400	50	458	467	1.97%	2.90%	4.81	−5.35%	−3.74%
Golden-11	400	58	456	468	2.63%	3.30%	4.64	−4.91%	−3.84%
Golden-11	400	67	454	469	3.30%	4.13%	4.20	−4.90%	−3.57%
Golden-11	400	80	451	468	3.77%	4.71%	3.87	−4.70%	−3.26%
Golden-12	484	33	535	545	1.87%	2.50%	4.38	−2.94%	−1.59%
Golden-12	484	35	537	547	1.86%	2.50%	3.40	−2.93%	−1.86%
Golden-12	484	38	535	547	2.24%	3.39%	4.15	−2.93%	−1.37%
Golden-12	484	41	537	550	2.42%	3.66%	3.67	−3.27%	−1.87%
Golden-12	484	44	535	552	3.18%	4.02%	4.11	−3.99%	−2.23%
Golden-12	484	49	533	550	3.19%	3.96%	4.01	−2.00%	−1.09%
Golden-12	484	54	535	551	2.99%	3.83%	4.74	−2.54%	−1.11%
Golden-12	484	61	538	552	2.60%	3.67%	4.16	−2.36%	−1.24%
Golden-12	484	70	546	552	1.10%	1.83%	3.76	−2.54%	−1.19%
Golden-12	484	81	546	557	2.01%	2.79%	4.52	−3.23%	−1.71%
Golden-12	484	97	560	566	1.07%	2.07%	3.98	−3.89%	−2.71%
Golden-13	253	17	552	553	0.18%	0.57%	4.43	−2.71%	−1.98%
Golden-13	253	19	549	552	0.55%	0.77%	4.79	−3.99%	−3.28%
Golden-13	253	20	548	549	0.18%	0.60%	4.84	−3.28%	−2.98%
Golden-13	253	22	548	549	0.18%	0.74%	2.80	−3.46%	−2.86%
Golden-13	253	23	548	551	0.55%	0.78%	3.73	−3.99%	−3.42%
Golden-13	253	26	542	544	0.37%	0.62%	3.69	−2.76%	−2.30%
Golden-13	253	29	540	543	0.56%	0.77%	3.64	−2.58%	−2.00%
Golden-13	253	32	543	545	0.37%	0.69%	4.75	−2.94%	−2.19%
Golden-13	253	37	545	550	0.92%	1.21%	3.84	−3.45%	−2.84%
Golden-13	253	43	553	559	1.08%	1.52%	3.24	−4.83%	−4.43%
Golden-13	253	51	560	565	0.89%	1.44%	3.90	−5.84%	−5.13%
Golden-14	321	22	692	698	0.87%	1.23%	4.23	−2.72%	−2.25%
Golden-14	321	23	688	692	0.58%	1.02%	4.27	−2.46%	−2.08%
Golden-14	321	25	678	683	0.74%	1.07%	3.87	−2.05%	−1.46%
Golden-14	321	27	676	683	1.04%	1.45%	4.35	−1.90%	−1.62%
Golden-14	321	30	678	684	0.88%	1.47%	3.94	−2.05%	−1.62%
Golden-14	321	33	682	687	0.73%	1.20%	3.68	−2.18%	−1.94%
Golden-14	321	36	687	689	0.29%	1.16%	3.35	−3.05%	−2.26%
Golden-14	321	41	690	695	0.72%	1.17%	5.46	−3.88%	−2.93%
Golden-14	321	46	694	699	0.72%	1.44%	3.99	−3.29%	−2.72%
Golden-14	321	54	699	706	1.00%	1.58%	3.83	−4.53%	−3.46%
Golden-14	321	65	703	713	1.42%	1.74%	3.55	−4.77%	−4.03%
Golden-15	397	27	842	850	0.95%	1.53%	5.14	−2.94%	−2.01%
Golden-15	397	29	843	854	1.30%	1.64%	5.01	−3.40%	−2.51%
Golden-15	397	31	837	846	1.08%	1.62%	4.64	−2.36%	−1.59%
Golden-15	397	34	838	851	1.55%	2.05%	3.95	−2.94%	−2.30%
Golden-15	397	37	845	858	1.54%	1.99%	4.09	−3.73%	−3.01%
Golden-15	397	40	849	859	1.18%	1.60%	2.90	−3.38%	−2.81%
Golden-15	397	45	853	864	1.29%	1.50%	4.39	−3.47%	−2.80%
Golden-15	397	50	851	863	1.41%	1.77%	3.04	−2.78%	−2.47%
Golden-15	397	57	850	859	1.06%	1.84%	3.98	−2.44%	−1.82%
Golden-15	397	67	855	870	1.75%	2.26%	3.74	−3.45%	−2.55%
Golden-15	397	80	857	874	1.98%	2.51%	3.32	−3.55%	−2.64%

Table 17
Detailed results for the Golden instances 16–20, as provided by Battarra et al. (2014).

Instance				Strong cluster constraints				Weak cluster constraints	
Set	n	N	opt.	Best	Avg. best GAP(%)	Avg. GAP(%)	Avg. CPU(s)	Best diff.	Avg. Diff.
Golden-16	481	35	1028	1038	0.97%	1.31%	4.24	−2.60%	−1.89%
Golden-16	481	37	1028	1037	0.88%	1.21%	3.66	−2.89%	−1.96%
Golden-16	481	41	1032	1039	0.68%	1.31%	4.43	−2.50%	−1.74%
Golden-16	481	44	1028	1042	1.36%	1.71%	4.53	−2.30%	−1.77%
Golden-16	481	49	1031	1044	1.26%	1.66%	5.56	−2.87%	−2.20%
Golden-16	481	54	1022	1038	1.57%	1.91%	4.39	−2.50%	−2.13%
Golden-16	481	61	1013	1035	2.17%	2.58%	3.85	−2.22%	−1.74%
Golden-16	481	69	1012	1035	2.27%	2.60%	3.86	−2.03%	−1.65%
Golden-16	481	81	1018	1043	2.46%	3.06%	2.93	−2.49%	−1.62%
Golden-16	481	97	1018	1048	2.95%	3.47%	3.16	−2.48%	−1.72%
Golden-17	241	17	418	421	0.72%	1.16%	3.44	−7.13%	−6.48%
Golden-17	241	18	419	421	0.48%	0.95%	2.77	−6.89%	−6.41%
Golden-17	241	19	422	424	0.47%	0.78%	5.01	−7.55%	−7.09%
Golden-17	241	21	425	427	0.47%	0.91%	3.68	−8.67%	−7.85%
Golden-17	241	22	424	426	0.47%	0.91%	4.59	−8.22%	−7.69%
Golden-17	241	25	418	421	0.72%	1.04%	3.70	−7.84%	−7.54%
Golden-17	241	27	414	416	0.48%	0.79%	2.29	−7.21%	−6.36%
Golden-17	241	31	421	422	0.24%	0.61%	2.33	−5.45%	−4.62%
Golden-17	241	35	417	418	0.24%	0.49%	3.34	−4.55%	−4.04%
Golden-17	241	41	412	**412**	**0.00%**	0.39%	4.96	−3.40%	−2.71%
Golden-17	241	49	414	416	0.48%	0.79%	3.45	−4.81%	−4.07%
Golden-18	301	21	592	599	1.18%	1.60%	4.13	−4.34%	−3.06%
Golden-18	301	22	594	602	1.35%	1.61%	4.89	−3.99%	−3.47%
Golden-18	301	24	592	601	1.52%	1.73%	5.09	−4.49%	−3.23%
Golden-18	301	26	590	595	0.85%	1.54%	3.41	−3.19%	−2.31%
Golden-18	301	28	577	582	0.87%	1.33%	4.42	−2.23%	−1.65%
Golden-18	301	31	578	583	0.87%	1.18%	4.09	−2.92%	−2.04%
Golden-18	301	34	582	585	0.52%	1.05%	4.38	−2.91%	−2.56%
Golden-18	301	38	586	592	1.02%	1.48%	2.32	−3.89%	−3.45%
Golden-18	301	43	594	599	0.84%	1.33%	4.50	−4.34%	−3.80%
Golden-18	301	51	601	605	0.67%	1.18%	3.29	−4.96%	−4.45%
Golden-18	301	61	599	602	0.50%	1.17%	3.73	−4.65%	−4.01%
Golden-19	361	25	925	936	1.19%	1.55%	5.06	−3.85%	−3.16%
Golden-19	361	26	924	935	1.19%	1.63%	4.84	−3.21%	−2.84%
Golden-19	361	28	808	818	1.24%	1.63%	4.36	−6.60%	−5.79%
Golden-19	361	31	811	822	1.36%	1.61%	4.41	−7.30%	−6.65%
Golden-19	361	33	797	806	1.13%	1.66%	4.96	−6.58%	−6.18%
Golden-19	361	37	799	809	1.25%	1.55%	4.51	−6.30%	−5.77%
Golden-19	361	41	789	796	0.89%	1.27%	4.03	−5.40%	−4.79%
Golden-19	361	46	788	794	0.76%	1.08%	3.13	−5.29%	−4.38%
Golden-19	361	52	800	807	0.88%	1.30%	3.43	−6.57%	−5.59%
Golden-19	361	61	807	812	0.62%	1.13%	3.09	−6.03%	−5.40%
Golden-19	361	73	810	814	0.49%	1.33%	3.00	−6.27%	−5.52%
Golden-20	421	29	1220	1231	0.90%	1.32%	4.78	−1.79%	−1.49%
Golden-20	421	31	1232	1239	0.57%	1.14%	4.30	−1.53%	−1.12%
Golden-20	421	33	1208	1219	0.91%	1.30%	4.37	−1.39%	−1.00%
Golden-20	421	36	1059	1073	1.32%	1.76%	5.44	−3.73%	−2.71%
Golden-20	421	39	1052	1063	1.05%	1.39%	5.13	−3.57%	−3.07%
Golden-20	421	43	1052	1062	0.95%	1.46%	3.37	−4.24%	−3.23%
Golden-20	421	47	1053	1065	1.14%	1.55%	3.90	−3.94%	−3.27%
Golden-20	421	53	1058	1071	1.23%	1.94%	3.41	−4.58%	−3.79%
Golden-20	421	61	1058	1072	1.32%	1.95%	3.82	−4.76%	−3.61%
Golden-20	421	71	1059	1076	1.61%	2.01%	2.80	−4.74%	−3.81%
Golden-20	421	85	1049	1062	1.24%	1.98%	4.38	−4.14%	−2.74%

References

Applegate, D., Bixby, R., Chvatal, V., Cook, W., 2006. Concorde tsp solver.

Barthélemy, T., 2012. A Bi-Objective Inventory Routing Problem with Periodic and Clustered Deliveries. Master's thesis, Universit de Nantes, France Ph.D. thesis.

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

Battarra, M., Erdoğan, G., Vigo, D., 2014. Exact algorithms for the clustered vehicle routing problem. Oper. Res. 62 (1), 58–71.

Bektas, T., Erdoğan, G., Røpke, S., 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Trans. Sci. 45 (3), 299–316.

Chisman, J.A., 1975. The clustered traveling salesman problem. Comput. Oper. Res. 2 (2), 115–119.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Manage. Sci. 6 (1), 80–91.

Defryn, C., Sörensen, K., 2015. A Two-Level Variable Neighbourhood Search for the Euclidean Clustered Vehicle Routing Problem. Research paper. University of Antwerp, Faculty of Applied Economics.

Ding, C., Cheng, Y., He, M., 2007. Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale TSPs. Tsinghua Sci. Technol. 12 (4), 459–465.

Expósito-Izquierdo, C., Rossi, A., Sevaux, M., 2016. A two-level solution approach to solve the clustered capacitated vehicle routing problem. Comput. Ind. Eng. 91, 274–289.

Fleszar, K., Hindi, K.S., 2002. New heuristics for one-dimensional bin-packing. Comput. Oper. Res. 29 (7), 821–839.

Garey, M.R., Johnson, D.S., 1978. "Strong" NP-Completeness results: motivation, examples, and implications. J. ACM (JACM) 25 (3), 499–508.

Ghiani, G., Improta, G., 2000. An efficient transformation of the generalized vehicle routing problem. Eur. J. Oper. Res. 122 (1), 11–17.

Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I.-M., 1998. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In: Fleet management and logistics. Springer, pp. 33–56.

Hansen, P., Mladenović, N., 2014. Variable neighborhood search. In: Burke, E.K., Kendall, G. (Eds.), Search methodologies. Springer, pp. 313–337.
Janssens, J., Van den Bergh, J., Sörensen, K., Cattrysse, D., 2015. Multi-objective microzone-based vehicle routing for courier companies: from tactical to operational planning. Eur. J. Oper. Res. 242 (1), 222–231.
Laporte, G., Palekar, U., et al., 2002. Some applications of the clustered travelling salesman problem. J. Oper. Res. Soc. 53 (9), 972–976.
Laporte, G., Potvin, J.-Y., Quilleret, F., 1997. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. J. Heuristics 2 (3), 187–200.
Lenstra, J.K., Kan, A., 1981. Complexity of vehicle routing and scheduling problems. Networks 11 (2), 221–227.
Li, F., Golden, B., Wasil, E., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Comput. Oper. Res. 34 (9), 2734–2742.
Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498–516.
Marc, A.H., Fuksz, L., Pop, P.C., Dănciulescu, D., 2015. A novel hybrid algorithm for solving the clustered vehicle routing problem. In: Onieva, E., Santos, I., Osaba, E., Quintián, H., Corchado, E. (Eds.), Hybrid Artificial Intelligent Systems. Springer, pp. 679–689.
Mestria, M., Ochi, L.S., de Lima Martins, S., 2013. GRASP with path relinking for the symmetric euclidean clustered traveling salesman problem. Comput. Oper. Res. 40 (12), 3218–3229.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.
Pop, P.C., Kara, I., Marc, A.H., 2012. New mathematical models of the generalized vehicle routing problem and extensions. Appl. Math. Model. 36 (1), 97–107.
Potvin, J.-Y., Guertin, F., 1996. The clustered traveling salesman problem: A genetic approach. In: Meta-Heuristics. Springer, pp. 619–631.
Sevaux, M., Sörensen, K., 2008. Hamiltonian paths in large clustered routing problems. In: Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME, 8, pp. 411–417.
Subramanian, A., 2012. Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems. University Federal Fluminense, Niterois (Brazil) Ph.D. thesis.
Vidal, T., Battarra, M., Subramanian, A., Erdoğan, G., 2015. Hybrid metaheuristics for the clustered vehicle routing problem. Comput. Oper. Res. 58, 87–99.