Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

Exact Algorithms for the Clustered Vehicle Routing Problem

Maria Battarra
School of Mathematics, University of Southampton, Highfield, Southampton, SO17 1BJ, UK, M.Battarra@soton.ac.uk

Güneş Erdoğan
School of Management, University of Southampton, Highfield, Southampton, SO17 1BJ, UK, G.Erdogan@soton.ac.uk

Daniele Vigo
DEI, University of Bologna, via Venezia 52, 47521 Cesena, Italy, daniele.vigo@unibo.it

This study presents new exact algorithms for the Clustered Vehicle Routing Problem (CluVRP). The CluVRP is a generalization of the Capacitated Vehicle Routing Problem (CVRP), in which the customers are grouped into *clusters*. As in the CVRP, all the customers must be visited exactly once, but a vehicle visiting one customer in a cluster must visit all the remaining customers therein before leaving it. Based on an exponential time preprocessing scheme, an integer programming formulation for the CluVRP is presented. The formulation is enhanced by a polynomial time graph reduction scheme. Two exact algorithms for the CluVRP, a Branch & Cut as well as a Branch & Cut & Price, are presented. The computational performance of the algorithms are tested on benchmark instances adapted from the Vehicle Routing Problem literature as well as real-world instances from a solid waste collection application.

Key words: Clusters, Vehicle Routing, Branch & Cut

1. Introduction

In this paper, we focus on the *Clustered Vehicle Routing Problem* (CluVRP), which is a generalization of the *Capacitated Vehicle Routing Problem* (CVRP) where customers are partitioned into *clusters*. We are given a complete undirected graph $G = (V, E)$, with demands $d_i, i \in V \setminus \{0\}$ associated with the vertices that represent the customers, and costs $c_{ij}, (i, j) \in E$ associated with the edges. The node 0 is the *depot*, where a homogeneous fleet of m vehicles is located. Each vehicle has a capacity Q and provides delivery service at the customer vertices. The customers' clusters are disjoint subsets of vertices with cardinality greater or equal to one. The set of all clusters is denoted as $\mathcal{C}$. A vehicle visiting one vertex in cluster $C \in \mathcal{C}$ must visit all vertices $i \in C$ before any other vertex. The objective of the CluVRP is to determine a set of m vehicle routes with minimum total cost, starting and ending at the depot, servicing each customer exactly once, without violating the vehicle capacity. The *Capacitated Vehicle Routing Problem* (CVRP) is a special case of the CluVRP, where all clusters are singletons. This relationship also proves that the CluVRP is NP-Hard.

The CluVRP has been introduced by Sevaux and Sörensen (2008) to model a real world application involving parcel deliveries or courier companies that use containers with stocked goods. The items to be delivered to customers in a specific area are stored into a container. All items in a container have to be delivered, so all the customers in a region or cluster must be served, before the content of an another container is distributed. In most applications, a single vehicle is expected to visit all the customers in a given area. Other applications of the CluVRP arise whenever clusters of customers prefer to be (or must be) served by the same vehicle. Typical examples of such situations are in the transportation of the elderly to recreation centers, where the customers prefer to be picked up together with friends or neighbors thus forming clusters. Similar constraints arise in the service of so-called "gated communities", which are residential or productive areas enclosed in walled enclaves for safety and protection reasons. Whenever more than one customer from a gated community requires service, these customers have to be visited by the same vehicle. Finally, CluVRP plays an important role as a practical tool to enforce route compactness where neighboring customers have to be served contiguously along the vehicle tours. In particular, in this paper we consider a case study where the CluVRP is used in the context of solid waste collection in urban areas. In this case, neighboring collection points are grouped into clusters that must be visited in sequence so as to enforce a sufficient compactness in the route shape.

The variants of the Vehicle Routing Problem (VRP) are amongst the most studied combinatorial optimization problems. For recent reviews of the VRP family, we refer the interested reader to the books by Toth and Vigo (2001) and Golden et al. (2008). The *Generalized Vehicle Routing Problem* (GVRP) is the Vehicle Routing Problem variant closest to the CluVRP. In the GVRP the customers are also grouped into clusters, but visiting a single customer in each cluster suffices to perform delivery. As we will discuss in the next section, for each CluVRP instance it is possible to derive an equivalent GVRP instance. However, since the number of vertices in the equivalent GVRP instance is considerably higher, such transformation is not of practical use for exact approaches.

In spite of the wide practical interest, the current literature on CluVRP is quite limited. The initial motivation was a large scale application presented by Sörensen et al. (2008) and Sevaux and Sörensen (2008). In subsequent work, Barthélemy et al. (2010) presented a Simulated Annealing algorithm based on inter-tour and intra-tour exchanges. To the best of our knowledge, there is no study that provides an exact algorithm for the CluVRP.

In this paper, we present a new compact and effective integer programming formulation for the CluVRP and two resulting exact solution algorithms. The formulation is based on a preprocessing algorithm that simplifies the problem by generating a polynomial number of columns in exponential time. This formulation is shown to dominate the adaptation of the CVRP two-index formulation and is the base of two exact algorithms. The first one is a Branch & Cut algorithm that uses all the columns, whereas the second algorithm is a Branch & Cut & Price algorithm in which the columns are generated dynamically. These two algorithms proved able to solve realistic size instances of CluVRP to optimality within limited computing times, with the Branch & Cut algorithm outperforming the Branch & Cut & Price algorithm.

The paper is organized as follows. In Section 2, we present two integer programming formulations and discuss their properties. We describe the associated preprocessing algorithm and graph reduction scheme in Section 3. We present the details of our Branch & Cut and Branch & Cut & Price algorithms in Section 4. We report our computational results on instances from the routing literature in Section 5. The details of the case study for the waste collection are provided in Section 6. Finally, in Section 7, we draw our conclusions.

2. Integer Programming Formulation

As stated in the introduction, to the best of our knowledge no specific formulation has been derived so far for the CluVRP. The only related work is a model proposed by Sevaux and Sörensen (2008) to find the optimal Hamiltonian path across a cluster. It is possible to transform an instance of CluVRP into an equivalent of GVRP in the following manner. For each cluster in the CluVRP instance we create a cluster in the GVRP instance and we add a vertex to the GVRP cluster for each possible entry vertex-exit vertex pair in the CluVRP cluster. This approach, however, results in a quadratic number of vertices. Hence, the resulting GVRP instances are usually out of the computational reach even for medium sized instances of CluVRP. Therefore, we propose in this section two integer programming formulations for our problem. The first one is a straightforward adaptation of a classical CVRP formulation, while the second one better exploits the cluster structure and forms the base of our exact approaches.

Following the problem definition given in the introduction, we now introduce the additional required notation. Let us define the set of edges connecting the vertices in set S to the vertices outside the set as $\delta(S) = \{(i,j) \in E : i \in S, j \notin S\}$, and the set of edges inside a vertex set $S \subseteq V$ as $E(S) = \{(i,j) \in E : i,j \in S\}$. We define $\bar{E} = \delta(C_1) \cup \delta(C_2) \cup ... \cup \delta(C_{|\mathcal{C}|})$ as the set of *inter-cluster edges*, and $\hat{E} = E \setminus \bar{E}$ as the set of *intra-cluster edges*. We denote the cluster to which vertex

i belongs as $\kappa(i)$. The total demand of cluster C is denoted as $d_C = \sum_{i\in C} d_i$, with all clusters requiring a single visit i.e. $d_C \leq Q, \forall C \in \mathcal{C}$. We write $\mathcal{S}$ to denote the set of all the subsets of V. We write $r(S)$ to denote the degree requirement of a vertex set S, computed as $r(S) = \lceil \frac{\sum_{i\in S} d_i}{Q} \rceil$.

A first model for the CluVRP is based on the two-index vehicle flow formulation for the CVRP (see, e.g., Toth and Vigo 2001), in which the edges connecting the vertices in a cluster to any vertex outside the cluster are constrained to be exactly 2. More precisely, x_{ij} variable assumes value 1 if edge (i,j) is in the solution, and 0 otherwise. The resulting formulation can be stated as:

$$
\begin{aligned}
(CluVRP1) \qquad & \text{minimize} \sum_{(i,j)\in E} c_{ij} x_{ij} && && (1)\\
& \text{subject to} \sum_{(i,j)\in\delta(i)} x_{ij} = 2, && \forall i \in V && (2)\\
& \sum_{(0,i)\in\delta(0)} x_{0i} = 2m, && && (3)\\
& \sum_{(i,j)\in\delta(S)} x_{ij} \geq 2r(S), && \forall S \in \mathcal{S}, && (4)\\
& \sum_{(i,j)\in\delta(C)} x_{ij} = 2, && \forall C \in \mathcal{C} && (5)\\
& x_{ij} \in \{0,1\}, && \forall \{i,j\} \in E : i \neq 0 \text{ or } |\kappa(j)| > 1, && (6)\\
& x_{0j} \in \{0,1,2\}, && \forall j \in V : |\kappa(j)| = 1. && (7)
\end{aligned}
$$

The objective function (1) minimizes the routing costs. Constraints (2) are the customers' degree constraints, and constraint (3) determines the degree of the depot. The so-called capacity-cut constraints (4) impose the capacity restrictions and also eliminate subtours from the solutions. Constraints (5) state the degree constraints for each cluster, representing the difference from the classical two-index formulation for the CVRP. Finally, the integrality of the variables is enforced by constraints (6) and (7).

Albeit simple, this model fails to exploit the special substructure of the clusters. We try to overcome this drawback by introducing a new formulation. To this end, let us observe that when the first and last vertices to be visited in a cluster $C \in \mathcal{C}, |C| \geq 2$ are $i, j \in C$, finding the optimum sequence to visit the remaining vertices in C amounts to determine a minimum cost Hamiltonian Path Problem (HPP). Once the optimal solutions of the HPPs $\forall i, j \in C : C \in \mathcal{C}$ are determined, a formulation can be constructed by using such information. Denoting the cost of the HPP that visits the cluster $C \in \mathcal{C}$ with endpoints $i, j \in C : i < j$ as $\hat{c}_{ij}$, let us define a new set of edge costs as c'_{ij} as $c'_{ij} = c_{ij}$ if $(i,j) \in \bar{E}$ and $c'_{ij} = \hat{c}_{ij}$ if $(i,j) \in \hat{E}$. Using these new edge costs, our problem is transformed into that of determining m vehicle routes, each respecting the capacity restrictions, visiting all the clusters exactly once, traversing a single edge in each cluster (i.e., a Hamiltonian path), and minimizing the routing cost. Figure 1 depicts an example of the transformation, with Figure 1a illustrating the solution of the HPP for each such clusters, and Figure 1b showing the selection of a single edge in each cluster with more than two customers.

We use $\mathcal{R}$ to denote the set of all the subsets of clusters in $\mathcal{C}$. The parameter $r(R) = \lceil \frac{\sum_{C \in R} d_C}{Q} \rceil$ is the degree requirement of cluster set $R \in \mathcal{R}$ based on the capacity restriction. Let x_{ij} be equal to 1 if the edge $(i,j) \in E$ is in the solution, and 0 otherwise (note that x_{0j} can assume value 2 if $|\kappa(j)| = 1$). The CluVRP can be formulated as follows:

$$(CluVRP2) \text{ minimize } \sum_{(i,j) \in E} c'_{ij} x_{ij} \tag{8}$$

$$\text{subject to } \sum_{j:(i,j) \in E(\kappa(i))} x_{ij} = \sum_{j:(i,j) \in \delta(\kappa(i))} x_{ij}, \quad \forall i \in V : |\kappa(i)| > 1, \tag{9}$$

$$\sum_{(i,j) \in \delta(C)} x_{ij} = 2, \quad \forall C \in \mathcal{C} \tag{10}$$

$$\sum_{i \in V \setminus \{0\}} x_{0i} = 2m, \tag{11}$$

$$\sum_{i \in R, j \notin R} x_{ij} \geq 2r(R), \quad \forall R \in \mathcal{R}, \tag{12}$$

$$x_{ij} \in \{0,1\}, \quad \forall \{i,j\} \in \bar{E} : i \neq 0 \text{ or } |\kappa(j)| > 1, \tag{13}$$

$$x_{0j} \in \{0,1,2\}, \quad \forall j \in V : |\kappa(j)| = 1. \tag{14}$$

$$x_{ij} \geq 0, \quad \forall \{i,j\} \in \hat{E}, \tag{15}$$

The objective function (8) aims at minimizing the routing cost. Constraints (9) impose that in cluster $C \in \mathcal{C} : |\kappa(i)| > 1$ if a vertex is used as an entry or exit point, then it must be connected to another vertex within the cluster. Constraints (10) enforce the degree of every cluster to be equal to 2. Constraints (11) require that m vehicle routes leave the depot. Constraints (12) are the capacity-cut constraints of CVRP adapted to CluVRP. Constraints (13) and (14) are the integrality constraints. We would like to stress the fact that the integrality constraints (13) are not stated for the intra-cluster edge variables, due to the fact that they are forced to integrality by the constraints (9) whenever the inter-cluster edge variables are integral.

We now give a formal proof that CluVRP2 yields a lower bound that is greater than or equal to that of CluVRP1. Given a formulation F, let us denote its linear programming relaxation as F^L the value of its optimal solution as $v(F)$, and the convex hull of its feasible solutions as $c(F)$. Furthermore, let us denote the objective function value of a solution x as $z(x)$. Finally, let us

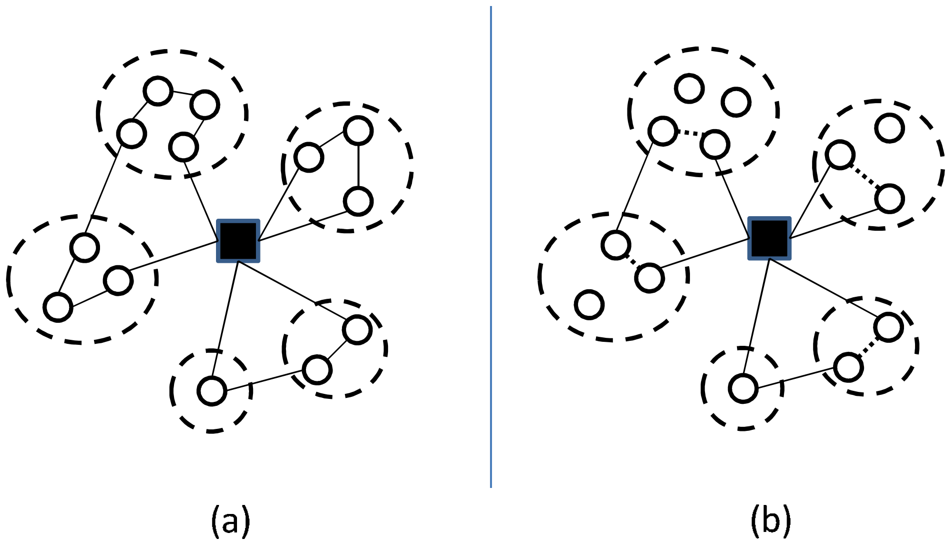

Figure 1 The transformation used by formulation *CluVRP2*.

denote the set of edges corresponding to a Hamiltonian path between two vertices i, j in cluster C as $P(i,j)$. We now show that given any solution $x^{**} \in c(CluVRP2^L)$, we can construct a solution $x^* \in c(CluVRP1^L)$ with the same objective function value. For each inter-cluster edge $(i,j) \in \bar{E}$, set $x^*_{ij} = x^{**}_{ij}$. Initialize $x^*_{ij} = 0$ for all intra-cluster edges $(i,j) \in \hat{E}$. For all $(p,q) \in \hat{E}$ and for all $(i,j) \in P(p,q)$, increment x^*_{ij} by x^{**}_{pq}.

Lemma 1. *Constraints* (9) *and* (10) *imply* $\sum_{(i,j)\in E(C)} x^{**}_{ij} = 1$.

Proof. Summing up constraints (9) for all $i \in V$ yields

$$2 \sum_{(i,j)\in E(C)} x^{**}_{ij} = \sum_{(i,j)\in\delta(C)} x^{**}_{ij}. \tag{16}$$

Replacing (10) for the right-hand-side and dividing by 2 yields $\sum_{(i,j)\in E(C)} x^{**}_{ij} = 1$. □

Proposition 1. *Degree constraints* (2) *are satisfied by* x^*.

Proof. Consider vertex i in cluster C, we have:

$$\sum_{j\in\delta(i)} x^*_{ij} = \sum_{j\in\delta(i)\cap\delta(C)} x^*_{ij} + \sum_{j\in\delta(i)\cap E(C)} x^*_{ij} \tag{17}$$

$$= \sum_{j\in\delta(i)\cap\delta(C)} x^{**}_{ij} + \sum_{j\in\delta(i)\cap E(C)} x^*_{ij} \tag{18}$$

$$= \sum_{j\in\delta(i)\cap\delta(C)} x^{**}_{ij} + \sum_{(p,q)\in E(C)\setminus\delta(i)} 2x^{**}_{pq} + \sum_{(p,q)\in E(C)\cap\delta(i)} x^{**}_{pq} \tag{19}$$

$$= \sum_{(p,q)\in E(C)\setminus\delta(i)} 2x^{**}_{pq} + \sum_{(p,q)\in E(C)\cap\delta(i)} 2x^{**}_{pq} \tag{20}$$

$$= 2\sum_{(p,q)\in E(C)} x^{**}_{pq} \tag{21}$$

$$= 2 \tag{22}$$

Equality (17) is the result of a partitioning of the edges in $\delta(i)$ as inter-cluster and intra-cluster. Equality (18) follows from the transformation from x^{**} to x^{*}. Equality (19) follows from the fact that vertex i has a degree of 2 on a Hamiltonian path in C if it is not an endpoint, and a degree of 1 if it is an endpoint. Equality (20) follows from the fact that the first and third terms in equality (19) are equal due to constraint set (9) of CluVRP2. Equality (21) is a regrouping of the edges. Finally, equality (22) follows from Lemma 1. □

LEMMA 2. *Given a set of customer vertices $S\subset V\setminus\{0\}$, with $\bar{C}$ being the minimal set of clusters covering S, i.e., $\forall i\in S, \exists C\in\bar{C}: i\in C$, the inequality $\sum_{(i,j)\in\delta(S)} x^{*}_{ij} \geq \sum_{(i,j)\in\delta(\bar{C})} x^{*}_{ij}$ holds.*

Proof. By induction on $|\bar{C}|$.
Base case: For $|\bar{C}|=1$, let $\bar{C}=\{C\}$. For $|C|\leq 2$, the inequality holds as an equality. For $|C|\geq 3$, we have:

$$\sum_{(i,j)\in\delta(S)} x^{*}_{ij} = \sum_{(i,j)\in\delta(C)} x^{*}_{ij} - \sum_{(i,j)\in\delta(C)\cap\delta(S)} x^{*}_{ij} + \sum_{(i,j)\in E(C)\cap\delta(S)} x^{*}_{ij} \tag{23}$$

$$= \sum_{(i,j)\in\delta(C)} x^{*}_{ij} - \sum_{(i,j)\in\delta(C)\cap\delta(S)} x^{**}_{ij} + \sum_{(i,j)\in E(C)\cap\delta(S)} x^{*}_{ij} \tag{24}$$

$$\geq \sum_{(i,j)\in\delta(C)} x^{*}_{ij} - \sum_{(i,j)\in\delta(C)\cap\delta(S)} x^{**}_{ij} + \sum_{(p,q)\in E(C)\cap\delta(S)} x^{**}_{pq} + \sum_{(p,q)\in E(C)\setminus\delta(S)} 2x^{**}_{pq} \tag{25}$$

$$\geq \sum_{(i,j)\in\delta(C)} x^{*}_{ij} - \sum_{(p,q)\in E(S)} 2x^{**}_{pq} + \sum_{(p,q)\in E(C)\setminus\delta(S)} 2x^{**}_{pq} \tag{26}$$

$$\geq \sum_{(i,j)\in\delta(C)} x^{*}_{ij} \tag{27}$$

Equality (23) states the relationship between $\sum_{(i,j)\in\delta(S)} x^{*}_{ij}$ and $\sum_{(i,j)\in\delta(C)} x^{*}_{ij}$, where the former is calculated by removing the inter-cluster edges in $\delta(C)\cap\delta(S)$ and by adding the intra-cluster edges belonging to $E(C)\cap\delta(S)$. Equality (24) follows from the transformation from x^{**} to x^{*}. Inequality (25) follows from the fact that a Hamiltonian path (p,q) in C with one endpoint in S and the other endpoint in $C\setminus S$ (i.e. $(p,q)\in E(C)\cap\delta(S)$) will cross between S and $C\setminus S$ at least

once, and at least twice if both p and q are in S or $C \setminus S$ (i.e. $(p,q) \in E(C) \setminus \delta(S)$). Inequality (26) is due to the fact that $\sum_{(i,j)\in\delta(C)\cap\delta(S)} x^{**}_{ij} = \sum_{(p,q)\in E(C)\cap\delta(S)} x^{**}_{pq} + \sum_{(p,q)\in E(S)} 2x^{**}_{pq}$, an implication of (9). Finally, (26) follows from the fact that $E(S) \subseteq E(C) \setminus \delta(S)$.
Inductive step: Assume that the inequality holds for $|\bar{C}| = k$. For $|\bar{C}| = k+1$, let us partition S into two nonempty, disjoint, and complementary subsets S_1 and S_2. Let $\bar{C}_1$ and $\bar{C}_2$ denote the minimal sets of clusters covering S_1 and S_2, respectively. Furthermore, let us choose S_1 and S_2 such that $\bar{C}_1 \cap \bar{C}_2 = \emptyset$. Then,

$$\sum_{(i,j)\in\delta(S)} x^*_{ij} = \sum_{(i,j)\in\delta(S_1)} x^*_{ij} + \sum_{(i,j)\in\delta(S_2)} x^*_{ij} - 2 \sum_{(i,j)\in\delta(S_1)\cap\delta(S_2)} x^*_{ij} \tag{28}$$

$$\geq \sum_{(i,j)\in\delta(\bar{C}_1)} x^*_{ij} + \sum_{(i,j)\in\delta(\bar{C}_2)} x^*_{ij} - 2 \sum_{(i,j)\in\delta(S_1)\cap\delta(S_2)} x^*_{ij} \tag{29}$$

$$\geq \sum_{(i,j)\in\delta(\bar{C}_1)} x^*_{ij} + \sum_{(i,j)\in\delta(\bar{C}_2)} x^*_{ij} - 2 \sum_{(i,j)\in\delta(\bar{C}_1)\cap\delta(\bar{C}_2)} x^*_{ij} \tag{30}$$

$$\geq \sum_{(i,j)\in\delta(\bar{C})} x^*_{ij} \tag{31}$$

Equality (28) states the relationship between the degrees of S, S_1, and S_2, where the degree of S is computed as the sum of the degrees of S_1 and S_2 minus twice the edges in the intersection of $\delta(S_1)$ and $\delta(S_2)$. Inequality (29) follows from the induction hypothesis. Inequality (30) follows from the fact that $\delta(S_1) \cap \delta(S_2) \subseteq \delta(\bar{C}_1) \cap \delta(\bar{C}_2)$. Finally, (31) is the result of the transformation used in (28), applied to $\bar{C}_1$ and $\bar{C}_2$. $\square$

Proposition 2. *Capacity constraints* (4) *are satisfied by* x^*.

Proof. We have $\sum_{(i,j)\in\delta(S)} x^*_{ij} \geq \sum_{(i,j)\in\delta(C)} x^*_{ij} = \sum_{(i,j)\in\delta(C)} x^{**}_{ij} \geq 2r(C) \geq 2r(S)$. The first inequality follows from Lemma 2. The equality is the result of the transformation from x^{**} to x^*. The next inequality is due to constraints (12). The last inequality is due to the fact that $r(C) \geq r(S)$. $\square$

Proposition 3. $v(CluVRP2^L) \geq v(CluVRP1^L)$.

Proof. Obviously, $z(x^*) = z(x^{**})$ and constraints (3) and (5) are honored by x^* due to the transformation and the constraints (10) and (11). Constraints (2) are satisfied due to Proposition 1. Finally, constraints (12) are satisfied because of Proposition 2. $\square$

As a consequence of the above propositions we have that formulation *CluVRP2* provides a better linear relaxation than *CluVRP1*. In addition, the preprocessing of the Hamiltonian Paths inside the clusters enable us to discard variables that will never be used, which we explain in detail in the next section.

3. Preprocessing

The definition of cost matrix $\hat{c}$ is clearly a crucial aspect of the practical usability of formulation *CluVRP2*. We now show how we compute the $\hat{c}_{ij}$ costs and provide a polynomial time graph reduction scheme that uses the Hamiltonian path distances and improves the computational performance.

3.1. Computing the Hamiltonian Path Distances

We now give a straightforward algorithm to compute all Hamiltonian path distances inside the clusters. To exploit existing efficient implementations, we solved this problem as an equivalent Traveling Salesman Problem (TSP) by means of the simple transformation obtained by setting $c_{ij} = -M$, where M is an arbitrarily large positive number, and solving the TSP over the vertices in cluster C. The pseudo-code of the algorithm is given below. Note that this approach is possible only if the code accepts negative costs. Otherwise one has to add a dummy node that is connected with zero cost arcs only to i and j and with M cost arcs to all other nodes.

Algorithm 1

```
For each clusters $C \in \mathcal{C} : |C| \geq 2$
  For each vertex pair $(i, j) \in E(C)$
    Store the value $c_{ij}$ and set $c_{ij} = -M$.
    Solve the TSP on $G_C = (C, E(C))$. Store the optimal solution as well as
      the optimal solution value $z^*_{ij}$.
    Set $\hat{c}_{ij} = z^*_{ij} + M$.
    Restore the value of $c_{ij}$.
```

Algorithm 1 requires the solution of $\sum_{C \in \mathcal{C}} |C|(|C| - 1)/2$ TSP instances. Although the number of instances looks prohibitive, our computational experience using CONCORDE (Applegate et al. 2001) to solve the required TSPs shows that instances including a limited number of vertices per cluster (e.g., 30 to 40 vertices) can be preprocessed within a reasonable computing time. Considering the largest GVRP instance from the literature ($|V|$=262) and aggregating the original clusters (that are small in size) to obtain larger clusters, our preprocessing algorithm defines all the Hamiltonian paths for about 12-14 clusters with on average 20 customers each within 10-15 minutes. In addition, this size of clusters is compatible with those required by most practical applications, as the waste collection problem which is considered in Section 6. Details about the computational performance of Algorithm 1 will be provided in Section 5.

3.2. Graph Reduction Scheme

Although *CluVRP2* is a compact formulation for the CluVRP, our initial computational experiments revealed that its LP relaxation may exploit the cluster structure to attain a weak lower bound, or a highly fractional solution that renders branching efforts ineffective. In Figure 2, a cluster with four vertices is depicted with four inter-cluster edge variables having values of 0.5 each. The intra-cluster edge variables may assume any of the three combinations of values depicted as well as all of their convex combinations. The solution of the LP relaxation naturally looks for the combination with the lowest cost, hence consisting of fractional solutions that allow splitting the inter-cluster edge variables as much as possible. This phenomenon occurs for clusters with more than 4 vertices, because of the number of intra-cluster edge variables exceeding the number of vertices in the cluster.

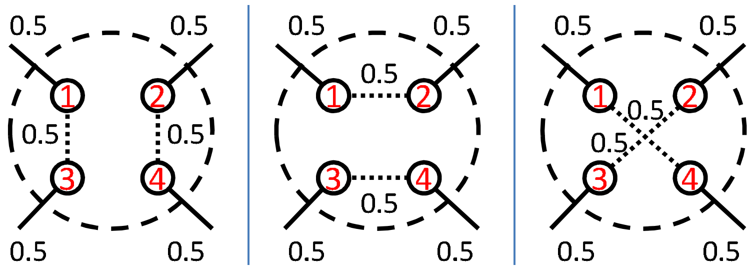

Figure 2 Possible fractional values of intra-cluster edge variables, for given fractional values of the inter-cluster edge variables.

A possible remedy to this shortcoming stems from the observation that many inter-cluster edges are never used in an optimal solution. Let us consider the example depicted in Figure 3 with two clusters having three vertices each, arranged in the form of equilateral triangles. In addition, vertices 2, 3, 4, and 5 are arranged in the form of a trapezoid (to leave alternative optimal solutions out of consideration) such that the distance between vertices 2 and 4 is slightly larger than that between vertices 3 and 5, i.e., $c_{24} = c_{35} + \epsilon$. Figure 3 enumerates all optimal solutions that connect these two clusters, for given entry points to the clusters from other clusters. Out of 9 inter-cluster edges, only a subset of cardinality four is used $\{(2,4),(2,5),(3,4),(3,5)\}$, and the remaining 5 inter-cluster edges may be discarded. We now state this observation as a formal result.

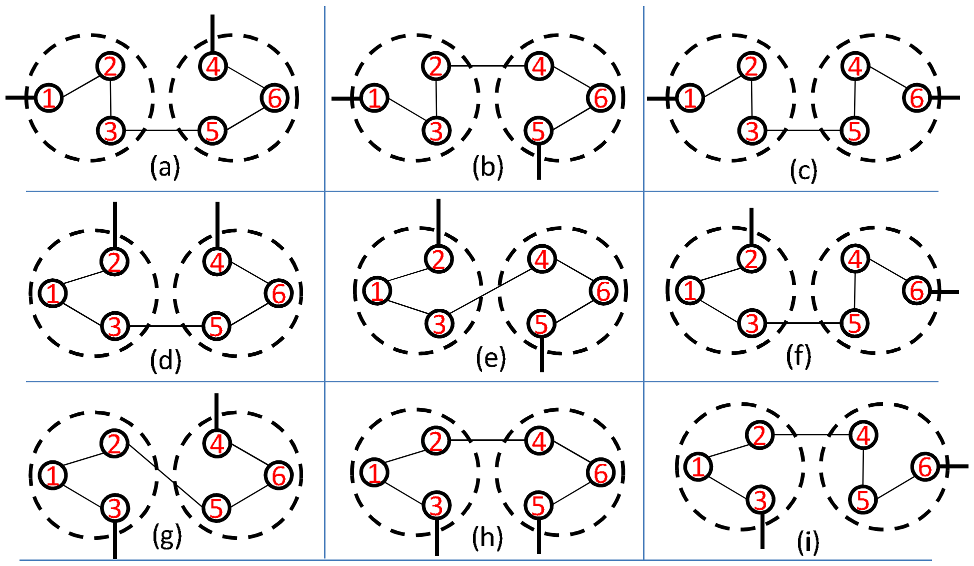

Figure 3 Edges used in optimal solutions connecting two clusters. Note that edges adjacent to vertices 1 and 6 are not used.

PROPOSITION 4. *An edge $(i,j) \in \bar{E}$ connecting two clusters $C_a, C_b \in \mathcal{C}$ with $i \in C_a$ and $j \in C_b$, will never be used in an optimal solution if for each $p \in C_a \setminus \{i\}, q \in C_b \setminus \{j\}$ there exists $(k,l) \in (\delta(C_a) \cap \delta(C_b)) \setminus \{(i,j)\}$ such that*

$$\hat{c}_{pk} + c_{kl} + \hat{c}_{lq} < \hat{c}_{pi} + c_{ij} + \hat{c}_{jq}. \tag{32}$$

Proof. Inequality (32) states that, for each tuple of entry points $\forall p \in C_a \setminus \{i\}, q \in C_b \setminus \{j\}$, there exists an edge (k,l) which connects the clusters C_a and C_b at a lower cost than (i,j). Clearly, in such a case the edge (i,j) will never be used. □

An implication of Proposition 4 is that an inter-cluster edge must be the optimal way of connecting two clusters for at least one pair of entry points. We now present an algorithm that marks the edges that are the optimal choice to connect two clusters for given entry points, and removes those that are never marked.

Algorithm 2

```
For each cluster pairs $C_a, C_b \in \mathcal{C} : C_a \neq C_b, |C_a| \geq 3, |C_b| \geq 3$
  Unmark all edges in $\delta(C_a) \cap \delta(C_b)$.
  Set $\alpha = +\infty$.
  For each vertex pair $p \in C_a, q \in C_b$
    For each vertex pair $i \in C_a, j \in C_b : i \neq p, j \neq q$
      If $\alpha > \hat{c}_{pi} + c_{ij} + \hat{c}_{jq}$ then set $\alpha = \hat{c}_{pi} + c_{ij} + \hat{c}_{jq}$.
    For each vertex pair $i \in C_a, j \in C_b : i \neq p, j \neq q$
      If $\hat{c}_{pi} + c_{ij} + \hat{c}_{jq} \leq \alpha$ then mark edge $(i, j)$.
  Delete edges in $\delta(C_a) \cap \delta(C_b)$ that are not marked
```

As the cardinality of a cluster grows, there are more vertices in the interior of the cluster that are less likely to be used as connection points, and there is a higher number of candidate edges for deletion. We would like to emphasize the fact that Algorithm 2 uses the results of Algorithm 1. The number of operations required by Algorithm 2 is $\sum_{C_a, C_b \in \mathcal{C}:2C_a \neq C_b} |C_a||C_b|(|C_a| - 1)(|C_b| - 1)$, which translates to a time complexity of $O(|V|^4)$ since $\sum_{C_a \in \mathcal{C}} |C_a| = |V| - 1$.

4. Exact approaches

In this section we present two different approaches for the exact solution of the CluVRP. The first one is a Branch & Cut algorithm and the second a Branch & Cut & Price algorithm, both based on the $CluVRP2$ formulation.

4.1. Branch & Cut

Our Branch & Cut algorithm is implemented through the CPLEX 12.5 integer programming solver. The linear relaxation of formulation $CluVRP2$ can be strengthened through various classes of valid inequalities. First of all, we used the CPLEX's embedded cut generation procedures. In addition, Bektaş et al. (2011) proved that all the CVRP inequalities from the literature can be extended to the GVRP, through shrinking every cluster into a single vertex and aggregating the edge variables betwen two clusters into a single variable. This result extends to the CluVRP, which allows us to utilize the Framed Capacity Inequalities and Hypotour Inequalities introduced by Augerat (1995), Generalized Large Multistar Inequalities presented by Achuthan et al. (1998), Homogeneous Multistar Inequalities developed by Letchford et al. (2002), and Strengthened Comb Inequalities

(Lysgaard et al. 2004). The Branch & Cut algorithm may benefit from these inequalities, as well as the dynamic separation of the capacity constraints. In particular, the separation procedures we adopted are those proposed by Lysgaard (2003).

During the extensive preliminary computational testing of our algorithm we observed that including the following set of constraints at the root node improved the performance of the algorithm:

$$\sum_{(i,j)\in\delta(C_1)\cap\delta(C_2)} x_{ij} \leq 1, \quad \forall C_1, C_2 \in \mathcal{C}, \tag{33}$$

These constraints, called Generalized Upper Bound (GUB) constraints hereafter, are a subset of the capacity-cut constraints and guarantee that no more than one edge is connecting a pair of clusters. In Section 5, the performance of these constraints, in comparison with CVRP inequalities and CPLEX internal cuts is analyzed in detail.

Finally, we implemented a simple heuristic used to initialize the Branch & Cut. A feasible solution is constructed by picking a minimum cost Hamiltonian path for each cluster with cardinality grater than 3 and setting the corresponding intra-cluster edge variable to 1. Then, the Branch & Cut is applied to this reduced problem to obtain a feasible solution.

4.2. Branch & Cut & Price

Although the overall number of variables is polynomial, the number of intra-cluster edges becomes considerably large as the cardinalities of the clusters increase. Consequently, the computational burden of the preprocessing strategy presented in Section 3 may turn out to be prohibitive. Therefore, we now present a column generation approach that aims at reducing the solution time for the HPPs by only considering those columns with negative reduced costs.

For clusters $C \in \mathcal{C}$ with $|C| \leq 3$, it is convenient to compute the $\hat{c}_{ij}, \forall (i,j) \in E(C)$ a priori, as described in Section 3. For the rest of the intra-cluster edges, the reduced costs associated with variables $x_{ij}, \forall (i,j) \in E(C), C \in \mathcal{C} : |C| \geq 4$ can be expressed as $\hat{c}_{ij} - u_i - u_j$ where u are the dual variables associated with constraints (9). The minimum reduced cost Hamiltonian path can be computed using the following formulation, originally proposed by Sevaux and Sörensen (2008). Let $\hat{x}_{ij}$ be a binary variable assuming value 1 if the edge $(i,j) \in C$ is traversed. Furthermore, let $\hat{y}_i$ be a binary variable equal to 1 if the node i is one of the endpoints of the Hamiltonian path in C. The resulting pricing problem is:

$$
\begin{aligned}
(SubP)\ \text{minimize} \quad & \sum_{(i,j)\in E(C)} c_{ij}\hat{x}_{ij} - \sum_{i\in C} u_i \hat{y}_i && && (34)\\
\text{subject to} \quad & \sum_{j:(i,j)\in E(C)} \hat{x}_{ij} = 2 - \hat{y}_i, && \forall i \in C && (35)\\
& \sum_{i\in C} \hat{y}_i = 2, && && (36)\\
& \sum_{(i,j)\in E(S)} \hat{x}_{ij} \le |S| - 1, && \forall S \subset C : |S| \ge 3 && (37)\\
& \hat{x}_{ij} \in \{0,1\}, && \forall (i,j) \in E(C) && (38)\\
& \hat{y}_i \in \{0,1\}, && \forall i \in C. && (39)
\end{aligned}
$$

The objective function (35) minimizes the reduced cost of the column to be generated. Constraints (35) impose a unit degree to the vertices that are selected as the endpoints, and a degree 2 for the remaining vertices of the cluster. Constraint (36) enforces the number of endpoints to be equal to 2. Constraints (37) are the well-known subtour elimination constraints, and (38) - (39) are the integrality constraints. This problem is $\mathcal{NP}$-hard as it contains the HPP as a special case. However, it can be solved in a reasonably short time by using a Branch & Cut algorithm for clusters of size up to 30, in which Constraints (37) are added dynamically in polynomial time.

To accelerate the pricing scheme, we exploit the fact that the reduced cost associated with each column depends on the endpoints of the cluster. This fact enables us to define a bounding strategy to minimize the number of HPPs solved at each iteration. Consider a minimum cost Hamiltonian path H from $i \in C$ to $j \in C$, which has a reduced cost of $\sum_{(k,l)\in H} c_{kl} - u_i - u_j$. Given the cost of the shortest Hamiltonian path H^* in C, the following relationship holds:

$$\sum_{(k,l)\in H^*} c_{kl} - u_i - u_j \le \sum_{(k,l)\in H} c_{kl} - u_i - u_j, \tag{40}$$

with the left hand side of the inequality yielding a lower bound for the reduced cost of H. Generalizing the idea to the whole cluster, a lower bound on the minimum of the reduced costs of Hamiltonian paths in cluster C is given by:

$$\sum_{(i,j)\in H^*} c_{ij} - \max_{p,q\in C: p\neq q} (u_p + u_q). \tag{41}$$

Since we are only interested in columns with negative reduced costs, we do not need to solve any HPPs associated with cluster C if the lower bound (41) is nonnegative. The use of this bound reduces the number of $SubP$ instances to be solved. Note that H^* can be determined as part of the initial column generation using $SubP$, and the two vertices in C having the maximum dual values can be found in $O(|C|)$ time.

Using the above mentioned results we implemented a Branch & Cut & Price algorithm. In particular, formulation $SubP$ is used for generating columns in conjunction with the bound (41), and the algorithm incorporates the same cuts implemented in the Branch & Cut approach. The performance of the resulting algorithms is discussed in the next section.

5. Computational Experiments

We extensively tested the Branch & Cut and Branch & Cut & Price algorithms on instances from the GVRP and instances we have adapted from the CVRP literature. The exact algorithms were based on CPLEX 12.5 solver framework and the computational testing was performed on an Intel i7 PC with 3.40 GHz, operating in Windows.

The first two sets of instances are the GVRP problems proposed by Bektaş et al. (2011) and available at `http://www.personal.soton.ac.uk/tb12v07/gvrp.html`. These problem classes, denoted as *GVRPθ2* and *GVRPθ3*, are adaptations of the CVRP instances from the CVRP-library (see `http://branchandcut.org/VRP/data/`). The experimental design to be presented below also follows that of Bektaş et al. (2011). In particular, a total of 73 CVRP instances are used as a starting point: namely all those from classes A, B, P and some of the M and G classes. For each CVRP instance the customer number, demands and coordinates are preserved.

The cardinality of the cluster set is set as $|\mathcal{C}| = \lceil n/\theta \rceil$, where $\theta \in \{2, 3\}$, transforming the CVRP instances ranging from 16 to 262 customers to CluVRP instances with 6 up to 131 small-sized clusters. The clustering technique is a *seed-based* algorithm: the $|\mathcal{C}|$ vertices farthest from each other are selected, and the remaining customers are assigned to the seeds using procedure CLUSTERING of Fischetti et al. (1997). For each cluster $C \in \mathcal{C}$, the demand d_C is scaled by a coefficient $\gamma = 1/|\mathcal{C}|$. This scaling was proposed to prevent the occurrence of short routes, as it may happen when $\gamma = 1$. The number of vehicles m is computed as the value of the Bin Packing solution obtained by the algorithm of Martello and Toth (1990). Euclidean distances were rounded to the nearest integer value.

Each instance is named using the notation $X - nY - kZ - CW - VJ$, where X corresponds to the original instance set (i.e., A, B, P, M, G), Y is the number of vertices, Z the number of vehicles in the CVRP instance, W the number of clusters, J the number of vehicles in the CluVRP instance. We grouped the instances obtained by setting $\theta = 2$ (*GVRPθ2*) and the instances obtained by setting $\theta = 3$ (*GVRPθ3*). Both sets of instances are challenging for CluVRP algorithms in terms of routing, since the resulting number of clusters is quite large (up to 131 in the instance $G - n262 - k25 - C131 - V12$). On the other hand the average number of customers per cluster is 2 and 3 for *GVRPθ2* and *GVRPθ3*, respectively. Hence, these instances are not challenging for the preprocessing algorithm described in Section 3.

The third set of CluVRP instances, denoted as *Clu-Golden*, is aimed at considering clusters with larger size. For this purpose, we used the 20 large-size CVRP instances proposed by Golden et al. (1998) (available at `http://www.rhsmith.umd.edu/faculty/bgolden/vrp_data.htm`) with 240 to 483 customers. We have adapted the CVRP instances to the CluVRP using the same procedure described above, with $\theta \in \{5, \dots, 15\}$. This results in an overall set of 220 instances (available upon request from the authors). The number of clusters varies from 16 to 97, and the cluster size varies from a single customer up to 71 customers. Finally, note that we have disregarded the route duration constraints whenever these were present in the original CVRP instances.

Our computational experiments were designed to evaluate the impact of the different components of our exact approaches: namely, $CluVRP2$, the effectiveness of cuts, and the effect of the graph reduction scheme.

Table 1 Comparison of the effectiveness of $CluVRP1$ and $CluVRP2$ models.

θ	Number of Inst.	Avg. % Dev.	CluVRP1	CluVRP2
5	20	0.05	14	18
6	20	0.13	14	17
7	20	0.06	15	18
8	20	0.07	15	19
9	20	0.07	18	19
10	20	0.09	18	20
11	20	0.09	18	20
12	20	0.10	19	20
13	20	0.16	19	20
14	20	0.19	18	20
15	20	0.20	18	20
Total:	220	0.11	186	211

First of all, we compared the performances of $CluVRP1$ and $CluVRP2$. To this end, we used either model as a base for the Branch & Cut algorithm of Section 4.1 and we applied them to the 220 *Clu-Golden* instances. Notably, this class of instances was observed to be more useful for the comparison of the two models, having both larger overall size and cardinality of clusters with respect to sets *GVRPθ1* and *GVRPθ2*. Table 1 reports the number of optimal solutions obtained by each algorithm within one hour of computing time, out of the 220 *Clu-Golden* instances. Note that in the $CluVRP2$ results, the graph reduction algorithm, the embedded CPLEX cuts, the CVRPSep Lib inequalities and the initial upper bound have been disabled, in order to provide a fair comparison with $CluVRP1$. The lines of the table report the average results for different values of θ, each including 20 instances. It can be easily observed that $CluVRP2$ is considerably more effective than $CluVRP1$, being able to solve a much larger number of instances (i.e., 211 versus 186, over 220). It should be also noted that the performance of $CluVRP2$ improves drastically when the number of clusters decreases (which corresponds to large values of θ), whereas $CluVRP1$ does not take a clear advantage by large-sized clusters. As a consequence, $CluVRP2$ is likely to successfully handle even larger instances with clusters of large cardinality, whereas $CluVRP1$ would not.

As proved in Section 2, the root node lower bound value of $CluVRP1$ is always worse than that of $CluVRP2$. However, the average deviation between the lower bounds produced by the two models is relatively limited, as presented column Avg. % Dev. of Table 1. Therefore, the superior performance of $CluVRP2$ is not only due to a stronger lower bound, but also to a better branching performance. The cost matrix c'_{ij} consists of heterogeneous cost coefficients (the terms correspond to either Hamiltonian paths or single edge distances) and the resulting objective function is less flat than in $CluVRP1$. This characteristic, in our opinion, allows $CluVRP2$ to obtain a better overall performance than $CluVRP1$.

Table 2 Comparison between the Branch & Cut and Branch & Cut & Price algorithms.

	GVRPθ2					GVRPθ3				
	B&C&P		B&C			B&C&P		B&C		
	Opt.	Avg. time	Opt.	Preproc. time	B&C time	Opt.	Avg. time	Opt.	Preproc. time	B&C time
A(27)	27	168.87	27	1.07	6.77	27	42.52	27	3.48	4.84
B(23)	23	16.62	23	1.05	2.22	23	7.69	23	4.39	4.99
P(24)	22	36.70	24	1.48	5.26	24	0.48	24	3.17	3.77
CMT/G(5)	1	113.70	3	3.00	8.50	2	157.25	4	8.00	25.44
Total(79):	73		77			76		78		
Avg:		80.31			4.90		21.72			1.27

We next compared the performances of the Branch & Cut and the Branch & Cut & Price algorithms, disabling in both algorithms the CPLEX embedded cuts, the CVRPSep inequalities, as well as the Upper Bound (UB) and Graph Reduction (GR) scheme (where applicable). The results of this test are shown in Table 2 and are based on instance sets *GVRPθ2* and *GVRPθ3*, because the Branch & Cut & Price was not capable of solving the large-sized instances in the *Clu-Golden* set. It can be seen that the Branch & Cut algorithm solved to optimality 6 instances more than the Branch & Cut & Price. Namely, four additional problems in the CMT/G group with $\theta = 2$ and two with $\theta = 3$ are solved by the Branch & Cut. In addition, the computing times for the Branch & Cut are remarkably smaller, even considering the time required for the preprocessing. Note that, although the clusters are small, increasing the value of θ from 2 to 3 (and the average cardinality of clusters from 2 to 3) more than doubles the preprocessing time. The average number of columns generated is 33.49 when $\theta = 2$, and it is 44.67 when $\theta = 3$.

A detailed analysis of preprocessing times is presented in Figure 4, where the minimum, average and maximum time, expressed in 10^4 CPU seconds, spent for the preprocessing of the *Clu-Golden*

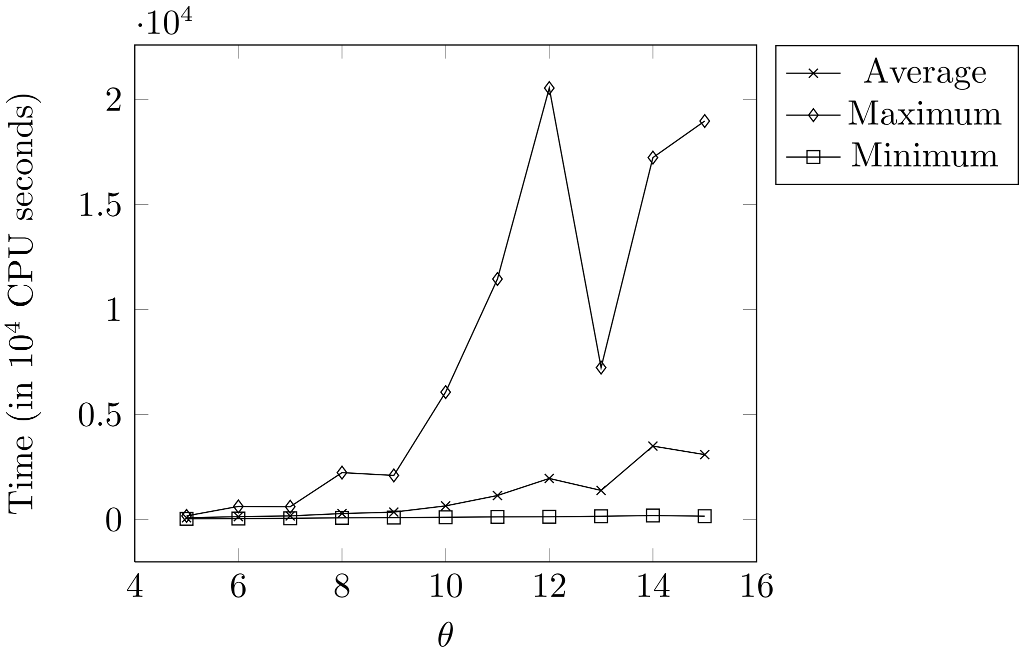

Figure 4 Preprocessing computing time.

instances with different θ values is presented. Note that there is a large variability among computing times due to the erratic behavior of the clustering algorithm, which sometimes generates subsets of nodes with similar cardinalities and other times results in clusters with quite variable cardinalities. When the latter happens, the largest clusters substantially increase the time required for preprocessing. The maximum preprocessing time within out experimentation has been observed to be 20541 seconds (i.e., less than 6 hours), which is acceptable since it allows to solve almost all the large-sized instances belonging to this class.

Table 3 Effects of additional cuts.

	No Cuts			CPLEX Cuts			CPLEX+CVRPsep			GUB		
θ	Avg. % Dev.	Time (sec.)	# Opt.	Avg. % Dev.	Time (sec.)	# Opt.	Avg. % Dev.	Time (sec.)	# Opt.	Avg. % Dev.	Time (sec.)	# Opt.
5	0.55	105.55	16	0.55	106.73	15	0.54	115.23	16	0.55	105.43	18
6	0.49	81.88	18	0.49	83.26	17	0.47	97.95	17	0.49	85.79	17
7	0.57	68.1	17	0.57	69.58	18	0.55	75.25	18	0.57	60	18
8	0.50	60.49	19	0.5	60.93	19	0.48	68.89	19	0.50	56.9	19
9	0.45	53.43	19	0.45	53.53	19	0.41	68.96	19	0.45	51.68	19
10	0.45	42.34	19	0.45	42.48	19	0.41	55.43	20	0.45	47.29	20
11	0.43	34.85	20	0.43	34.64	20	0.39	47.1	20	0.43	33.39	20
12	0.53	45.23	20	0.53	41.08	20	0.46	53.86	20	0.53	34.82	20
13	0.46	30.13	20	0.46	30.07	20	0.39	43.83	20	0.46	25.47	20
14	0.44	24.95	20	0.44	24.89	20	0.39	39.88	20	0.44	25.13	20
15	0.36	19.21	20	0.36	19.16	20	0.32	31.97	20	0.36	21.26	20
Avg.	0.48	51.47	18.91	0.48	51.49	18.82	0.44	63.49	19.00	0.48	49.74	19.18
Avg.>9	0.45	32.79	19.83	0.45	32.05	19.83	0.39	45.35	20.00	0.45	31.23	20.00

We also performed a number of tests to assess the contribution of additional cuts: we included in turn in our Branch & Cut algorithm the inequalities from the CPLEX cuts, the CVRPSep Lib inequalities (as described in Section 4) and the GUB Constraints (33). For each experiment, Table 3 reports the average percentage deviation from the best known solution at the root node, the corresponding computing time (in seconds), and the number of optimal solutions obtained within one hour of computing time by the Branch & Cut algorithm. It is interesting to note that, at least for smaller θ values, neither CPLEX cuts nor CVRPsep cuts are substantially improving the performance of the algorithm. We attribute this to the large size of the instance and the large number of variables involved, CPLEX and CVRPsep cuts increase of about 10% the average root node computing time and reduce the root average deviation by only 0.04% (denoted by the line marked Avg that reports the average values over all instances). On the other hand the GUB constraints reduce both the average computing time and allow for solving to optimality two more instances that the CPLEX+CVRPsep algorithm. Therefore, we opted to include only GUB constraints in the standard version of our algorithm. However, as previously mentioned, the benefit of using CPLEX and CVRPsep cuts is more noticeable for larger θ values as reported by the last line (marked as Avg.>9) which gives the average of the results for instances with θ larger than 9. We note that in this case the root node deviation is better than that of GUB with a moderate increase of the average computing time. As a consequence, when the average size of clusters is relatively larger the exact approach may profitably exploit the presence of more sophisticate cuts in addition to the GUBs.

Table 4 Performance of the upper bound

θ	Avg. Dev. (%)	No Feasible Solution	Avg. Time (sec.)
5	6.75	1	633.12
6	7.56	1	580.00
7	8.55	0	427.90
8	8.89	0	306.54
9	9.65	0	172.36
10	9.60	0	66.41
11	9.74	0	45.57
12	9.80	0	35.29
13	9.89	0	21.90
14	9.87	0	16.14
15	10.01	0	26.39
Avg.	9.12		154.70

Finally, in our experiments, we have observed that the Branch & Cut algorithm benefits from the presence of an initial upper bound value. As described in Section 4.1, this is computed by picking a minimum cost Hamiltonian path for each cluster with cardinality grater than 3, setting the corresponding intra-cluster edge variable to 1, and applying the Branch & Cut on this reduced problem. The performance of such a simple upper bound is presented in Table 4, where the deviation from the best known solutions, the number of instances for which the heuristic was able to find a feasible solution, and the average computing time are reported for the *Clu-Golden* instances. Contrary to our expectations, the most convenient Hamiltonian paths do not lead to high quality routing solutions: the average optimality gap for the upper bound is in fact as large as about 10% and the average computing time is 154.70 seconds. Moreover, in two instances the Branch & Cut was not capable of finding any integer solution, within the one hour time limit. This emphasizes the need for more sophisticated heuristics for the CluVRP.

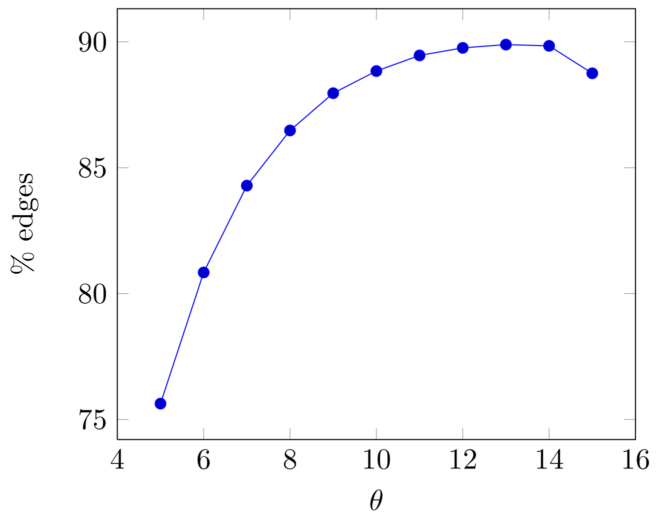

Figure 5 Graph reduction effectiveness: % of edges disregarded by varying θ.

Table 5 Performance of the upper bound and the graph reduction scheme.

		B&C							
		basic		GR		UB		GR+UB	
θ	Opt.	Avg. Time	# Opt.	Avg. Time	# Opt.	Avg. Time	# Opt.	Avg. Time	# Opt.
5	17	364.72	18	431.97	18	492.09	17	363.00	17
6	16	198.59	17	136.15	19	185.19	17	86.65	19
7	17	110.40	18	143.08	19	202.39	17	109.71	19
8	19	224.76	19	90.10	19	184.63	19	93.86	19
9	19	117.28	19	87.92	19	140.70	19	92.18	19
10	19	42.83	20	62.14	20	84.28	19	48.82	20
11	20	129.71	20	80.33	20	242.03	20	72.23	20
12	20	96.04	20	45.29	20	83.09	20	48.71	20
13	20	56.19	20	42.24	20	64.50	20	40.20	20
14	20	93.43	20	41.12	20	101.03	20	39.18	20
15	20	44.50	20	24.73	20	68.45	20	23.09	20
Avg.	207	130.26	211	102.38	214	162.97	208	88.66	213

Finally, we analyze the effect of the GR and the UB on the performance of the Branch & Cut algorithm. Figure 5 depicts the percentage number of edges discarded by the GR, by varying θ.

Remarkably, a maximum 89.90% of the edges are discarded, when $\theta = 13$. When the size of the clusters are increased, the GR scheme proves to be more effective, but the % of edges disregarded stabilizes when $\theta > 10$.

Table 5 reports the average computing times in seconds for the Branch & Cut in which neither upper bound nor the graph reduction scheme is applied (column marked *Basic*). The following columns present the average computing times when only the UB is used, when only the GR is applied, and when both UB and GR are utilized, respectively. The UB does not improve the root node solution time, but allows for solving an extra instance to optimality. The GR is effective in reducing the average root time by roughly 20%, and provides three additional optimal solutions, The combined use of the GR and UB reduces drastically the average root node computing time (approximately 33%) and 213 instances are solved to optimality. These results confirm the importance of the GR scheme, and provide motivation for further research for high quality heuristics.

6. Case Study

The exact approach for CluVRP has been tested on a real-world application of urban garbage collection, a challenging application of street routing (see, e.g., Bodin et al. 2003 and Sahoo et al. 2005). In this type of problems, practitioners place a strong emphasis on the so-called route *visual attractiveness*. Although this concept is quite intuitive, it is not easy to be defined quantitatively and mainly refers to the fact that the routes serving a given area "should not cross each other too much" (see, e.g., Poot et al. 2002). As illustrated by Figure 6, heuristic solutions may be shorter but much less compact (thus less visually attractive) than typical manually constructed solutions, which are often built starting from a partition of the service area into zones assigned to different vehicles. Benefits of visually attractive routes are manifold in practical applications and they often greatly compensate a smaller efficiency in terms of variable costs. First of all, they are more likely to be accepted by planners and drivers, being closer to their usual practice. Second, they often require a much smaller effort to be implemented, reducing the time required to instruct the drivers about the routes. Finally, they may be more stable in performance along time since they refer to more homogeneous areas in terms of traffic condition and are subject to continuous refinement by exploiting driver familiarity with the area served by the route.

Introducing visual attractiveness into route building procedures is a non-trivial task from the algorithmic point of view, due to both the fuzzy and subjective definition of the property and the great variability of situations that have to be faced by considering various urban shapes and density of collection points. In the rather limited literature on this subject, it is often achieved by using algorithms which can incorporate some route compactness mechanisms coupled by manual editing of the routes into GIS-based decision support systems (see, e.g., Poot et al. 2002).

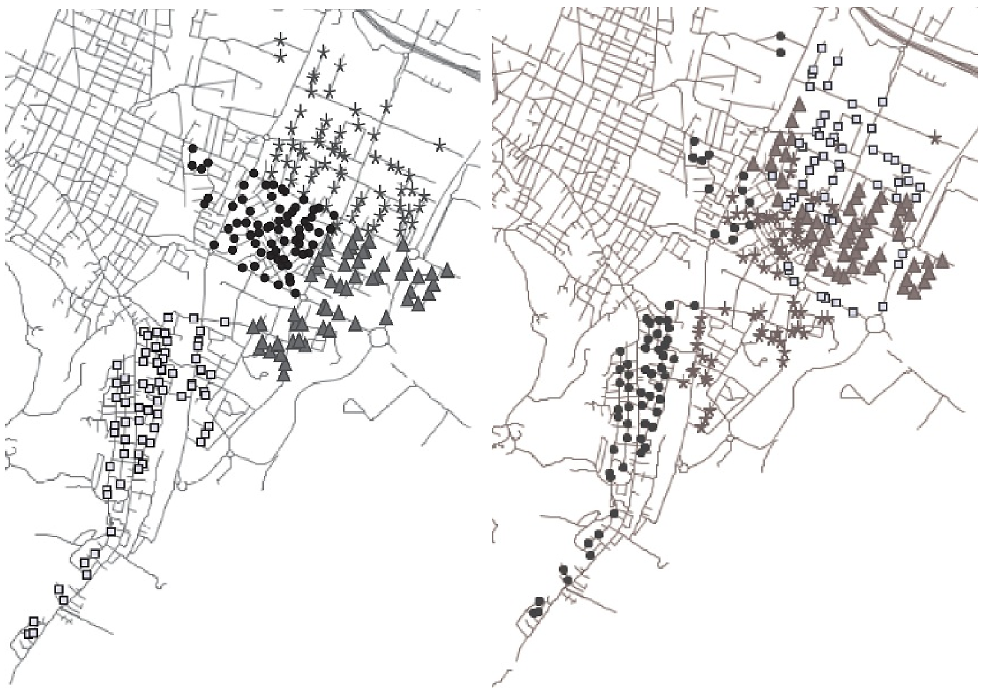

Figure 6 Comparison between hand-made and heuristic solutions of a real-world waste collection problem with four routes. The hand-made solution (left) is about 4% longer but more compact than the heuristic one (right).

A practical way to keep route compactness and to allow users to "customize" this aspect is to let the customer cluster the collection points. In this case, by varying the average size of the clusters the user may easily trade-off between route compactness and potential cost of sub-optimality associated with territory-based routes. Within these settings the possibility of solving CluVRP problems to optimality represents a very interesting opportunity for achieving cost-effective solutions which also incorporate a good degree of visual attractiveness. To evaluate such an opportunity in more depth, we conducted some experiments on a real-world dataset obtained from the solid waste collection in a town in the north of Italy. The data refers to a district of the town where 456 large street bins are located at 385 collection points. To make the evaluation of visual attractiveness easier, we used Euclidean distances rather than road distances between points, since the latter may quite strongly distort the shape of routes because of one way and turning restrictions. The vehicles available for the service have a capacity of 250 bins and are located at a depot that is considerably far away from the district. The company provided us with instances consisting of several clustering patterns: the maximum cardinality of the clusters varies between 5 and 30 bins, corresponding to an average of 4 to 23 collection points. For each clustering pattern, we compared the solutions obtained by our exact approach with those obtained by a commercial solver used by the company.

The results are summarized in Table 6. For each clustering pattern, the table reports the number of clusters and the corresponding maximum and average sizes. Columns marked "Heuristic solution" include the results obtained by the commercial heuristic when applied to the instance: the table reports the solution value, the percentage deviation with respect to the optimal solution value and the computing time in seconds. The last six columns of the table present the results of the exact approach, including the time for the computation of the Hamiltonian paths, the results of the upper bound computation, the optimal solution value, and the total computing time in seconds.

The exact algorithm is capable of solving efficiently instances from real world applications with reasonable size. In particular, if we consider the cases with cluster size between 10 and 20, which provide a very good compromise between quality and compactness, the computing time of the exact approach is comparable or smaller than the heuristic and obtains solutions which are much better in quality. In addition, this experiment indicates the possibility of handling even larger instances within the time limit of one hour of computing time provided the total number of clusters is smaller than 80-100, thus resulting in a number of customers which can be 4-5 times larger than those in the current real-world problem. On the average, the UB is improving the heuristic solutions of 9.07%, the exact approach improves the same result of 12.93%.

Table 6 Results or real-world instances from solid waste collection.

							Exact solution					
	Clusters			Heuristic solution			Preproc.	Initial UB			Optimal	Total
Problem	Number	Max size	Avg size	Value	%	Time	Time	Value	%	Time	Value	Time
R05	100	5	4	63397	116.5	356.0	40.0	55569	102.1	618.8	54424	2328.4
R08	63	8	6	66548	117.6	238.0	68.0	57775	102.1	20.6	56576	106.3
R10	50	10	8	65038	115.0	197.0	89.0	58684	103.7	62.8	56578	342.2
R15	34	15	11	61585	110.6	148.0	181.0	58656	105.3	31.8	55694	43.6
R20	25	20	15	61615	108.8	124.0	273.0	58995	104.2	44.1	56612	52.4
R25	20	25	19	63566	112.3	107.0	364.0	58284	103.0	0.2	56609	39.0
R30	17	30	23	60739	109.7	90.0	446.0	57901	104.6	46.0	55347	47.5
Average					112.9	180.0	208.7		103.6	117.8		422.8

The results of the exact approach are quite stable in terms of quality when the number of clusters is varied, as further illustrated by Figure 7 where the value of the optimal solution corresponding to the different clustering patterns is reported. It can be observed that the solution value does not worsen significantly when the size of the clusters increase. On the other hand when the cluster size is relatively larger, solution compactness is greatly increased as shown by Figure 6 which shows an example of the exact solution for the case labeled R08 (where each label is the cluster id associated to a garbage collection point).

7. Conclusion

In this study, we have developed an integer programming formulation for the CluVRP, based on a preprocessing algorithm that simplifies the problem structure. The preprocessing algorithm

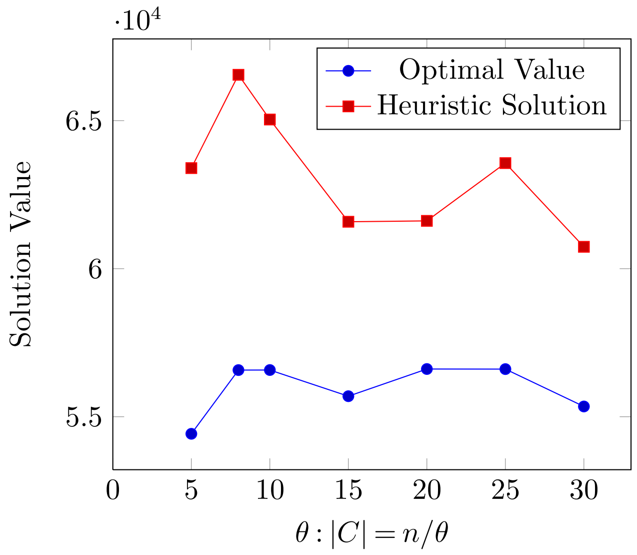

Figure 7 Solution value by varying the number of clusters.

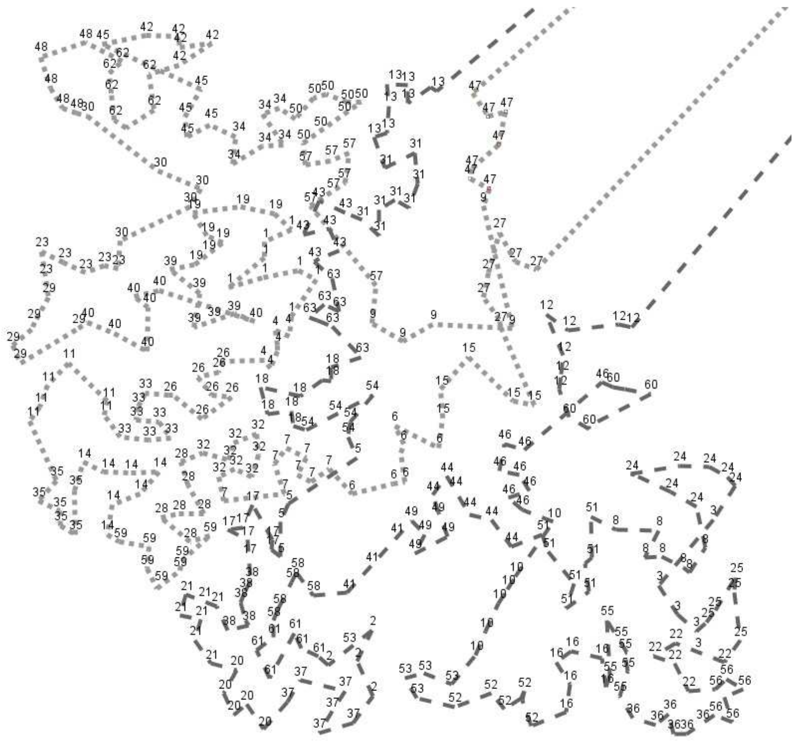

Figure 8 Example of solution in the garbage collection application.

requires the solution of a quadratic number of small TSP instances, which were observed to be within computational reach. The result of the preprocessing algorithm can be further utilized to discard variables through an effective graph reduction scheme. Two exact algorithms are presented, a Branch & Cut as well as a Branch & Cut & Price, into which the valid CVRP inequalities from the literature are incorporated. The results achieved by the Branch & Cut algorithm has been extensively tested on test instances from literature benchmarks and on real-world data from a garbage collection application. The Branch & Cut algorithm was observed to be capable of solving large-sized instances within reasonable computing time and outperformed the results obtained by alternative formulations not benefiting from the preprocessing algorithm and the graph reduction scheme. As shown in our case study, the exact approaches proved able to produce solutions whose quality is considerably superior with respect to commercial heuristics in relatively short computing times. Future research may be devoted to develop effective metaheuristic algorithms for the CluVRP.

Acknowledgments

The authors would like to acknowledge CORMSIS and Ministero dell'Istruzione, dell'Universitá e della Ricerca (Italy) for partially funding this project. Thanks are also due to an anonymous company for providing us the real-world data and to Mattia Manfroni for his help in the computational testing for the case study.

References

N.R. Achuthan, L. Caccetta, and S.P. Hill. Capacitated vehicle routing problem: some new cutting planes. *Asia Pacific Journal of Operational Research*, 15:109–123, 1998.

D. Applegate, R. Bixby, V. Chvàtal, and W. Cook. Concorde tsp solver, 2001. URL `http://www.tsp.gatech.edu/concorde/index.html`.

P. Augerat. *Approche Polyedrale du Probleme de Tournees de Vehicules*. PhD thesis, Institut National Polytechnique de Grenoble, 1995.

T. Barthélemy, A. Rossi, M. Sevaux, and K. Sörensen. Metaheuristic approach for the clustered vrp. In *EU/ME 2010 - 10th anniversary of the metaheuristic community*, Lorient, France, June 2010.

T. Bektaş, G. Erdoğan, and S. Ropke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45:299–316, 2011.

L. Bodin, V. Maniezzo, and A. Mingozzi. Street routing and scheduling problems. In Randolph W. Hall, editor, *Handbook of Transportation Science*, volume 56 of *International Series in Operations Research & Management Science*, pages 413–449. Springer US, 2003. ISBN 978-1-4020-7246-8. doi: 10.1007/0-306-48058-1_12.

M. Fischetti, J.-J. Salazar-González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

B.L. Golden, E.A. Wasil, J. P. Kelly, and I.-M. Chao. Metaheuristics in vehicle routing. In *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston, 1998.

B.L. Golden, S. Raghavan, and E.A. Wasil, editors. *The vehicle routing problem: latest advances and new challenges*. Springer, New York, NY, 2008.

A.N. Letchford, R.W. Eglese, and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94:21–40, 2002.

J. Lysgaard. Cvrpsep: A package of separation routines for the capacitated vehicle routing problem. Working paper, 2003.

J. Lysgaard, A.N. Letchford, and R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.

S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley, 1990.

A. Poot, G. Kant, and A.P.M. Wagelmans. A savings based method for real-life vehicle routing problems. *Journal of the Operational Research Society*, 53(1):pp. 57–68, 2002.

S. Sahoo, S. Kim, B.-I. Kim, B. Kraas, and A.Jr. Popov. Routing optimization for waste management. *Interfaces*, 35(1):24–36, 2005.

M. Sevaux and K. Sörensen. Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France, October 2008.

K. Sörensen, J. Van den Bergh, D. Cattrysse, and M. Sevaux. A multiobjective distributionproblem for parcel delivery at tnt. In *Invited Talk at the International Workshop on Vehicle Routing in Practice, VIP'08*, Oslo, Norway, June 2008.

P. Toth and D. Vigo. *The vehicle routing problem.* SIAM, Philadelphia, PA, 2001.