

Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem

Tolga Bektaş*

School of Management, University of Southampton, Highfield, Southampton, SO17 1BJ, UK
T.Bektas@soton.ac.uk

Güneş Erdoğan

Faculty of Economics and Administrative Sciences, Ozyegin University, Kuşbaşı Cad. No: 2, Altunizade, Üsküdar, Istanbul, 34662, Turkey
gunes.erdogan@ozyegin.edu.tr

Stefan Ropke

Department of Transport, Technical University of Denmark, Bygningstorvet 115, 2800 Kgs. Lyngby, Denmark
sr@transport.dtu.dk

The Generalized Vehicle Routing Problem (GVRP) consists of finding a set of routes for a number of vehicles with limited capacities on a graph with the vertices partitioned into clusters with given demands such that the total cost of travel is minimized and all demands are met. This paper offers four new integer linear programming formulations for the GVRP, two based on multicommodity flow and the other two based on exponential sets of inequalities. Branch-and-cut algorithms are proposed for the latter two. Computational results on a large set of instances are presented.

Key words: Generalized Vehicle Routing; Integer Programming; Branch-and-Cut.

1. Introduction

In this paper, we are concerned with the *Generalized Vehicle Routing Problem* (GVRP) that consists of finding a set of routes for a number of vehicles with limited capacities on a graph with the vertices partitioned into clusters with given demands such that exactly one vertex from each cluster is visited, the total cost of travel is minimized and all demands are met. The GVRP has applications mainly in distribution network planning, and to some extent in telecommunications network design. Some immediate application domains are listed below:

1. *Routing vessels in maritime transportation:* Given a number of regions with a number of ports located in each, if the distribution plan is such that the ships should deliver the goods to only a single port in each region (thence the goods can be distributed within the region), then the corresponding routing problem can be modeled as a GVRP where the regions correspond to the clusters and the fleet of vessels corresponds to the vehicle set.

*Corresponding author.

2. *Healthcare logistics*: A specific application of the GVRP arises when a number of districts, each encompassing a number of municipalities need to be provided with pharmaceutical products and it suffices to deliver a whole batch of products to one municipality within each district. In this case, the distribution problem corresponds to a GVRP for which the districts correspond to the clusters and the medical distribution team corresponds to the the vehicle set.

3. *Urban waste collection problem*: An application of the GVRP to solve an urban waste collection problem is reported by Bautista et al. (2008), which consists of devising routes for a number of vehicles which are used to pick-up urban waste and deliver it to a refuse dump, an incinerator or a recycling plant at minimum transportation cost.

4. *Survivable telecommunication network design*: Given a centralized telecommunications infrastructure with a central server and clusters of computers where the design is to be done such that exactly one computer in each cluster is connected with the central server, and that there exist exactly two edge disconnected paths from the central vertex to each cluster. The survivability of the network is ensured by the two edge disconnected paths where one would be activated in case the other fails. Clearly, any feasible solution to the GVRP can be used to implement such a design where the central server is the depot.

The GVRP is a generalization of the *Capacitated Vehicle Routing Problem* (CVRP) which lies at the heart of distribution management. The literature on the CVRP is quite rich and we refer the reader to Cordeau and Laporte (2006), Cordeau et al. (2007), Laporte (2007) for overviews of the recent progress on this problem. The GVRP also generalizes the *Generalized Traveling Salesman Problem* which has attracted considerable attention (see, e.g., Fischetti et al. 1995, 1997).

The GVRP is \mathcal{NP} -Hard as it contains the CVRP as a special case. The existing literature on the GVRP is quite scarce. The earliest published article on the GVRP is , to the best of our knowledge, Ghiani and Improta (2000), who describe a transformation of the problem into another \mathcal{NP} -Hard problem, namely the *Capacitated Arc Routing Problem* (CARP), which therefore enables one to utilize the available algorithms for the latter in solving the former. In an unpublished work by Kara and Bektaş (2003), a polynomial sized formulation is proposed for the GVRP incorporating additional restrictions on the load carried by each vehicle. This is an assignment based formulation using the well-known Miller-Tucker-Zemlin (MTZ) (Miller et al. 1960) constraints for the *Traveling Salesman Problem* adapted for the CVRP (see Kara et al. 2003).

A very recent work by Bautista et al. (2008) studies a special case of the GVRP derived from a waste collection application where each cluster contains at most two vertices. The authors propose a three-index directed formulation to model the problem based on exponential number of

inequalities, and also describe a polynomial-size variation of this model using MTZ-like constraints. The application as reported by these authors is initially put forward as a CARP, but the model is based on its transformation to a node routing problem which, has the form of a GVRP. The authors describe a number of heuristic solution procedures, including two constructive heuristics, a local search method and an ants heuristic to solve their practical instances, but no computational experience with the proposed formulation is reported in their paper.

The aim of this paper is to develop an efficient exact solution algorithm for the GVRP. The main contributions of the paper are as follows: 1) we present four new formulations for the GVRP, 2) we compare the four formulations both analytically and empirically, 3) we present a simple metaheuristic and preprocessing algorithm for the GVRP, 4) we propose a new data set for the GVRP containing 158 instances, 5) we show that instances with up to 121 nodes and 41 clusters are within reach of a branch-and-cut algorithm based on the best of the four formulations.

A formal description of the problem and four different formulations are presented in Section 2. Section 3 describes the general framework of the branch-and-cut algorithm devised to solve two of the four formulations. An effective preprocessing technique that is able to reduce the size of some GVRP instances is presented in Section 4, which is followed by a description of a metaheuristic in Section 5 used to calculate upper bounds for the problem. Section 6 presents the results of an extensive set of computational experiments in comparing and testing the formulations. Conclusions are stated in Section 7.

2. Formulations

The formal definition of the problem is given as follows: the GVRP is defined on a graph $G = (V, E)$ with $V = \{0, 1, \dots, n\}$ as the set of vertices. Vertex 0 corresponds to the depot and the remaining vertices correspond to customers. V is partitioned into (nonempty and disjoint) subsets called clusters as $\{C_0, C_1, \dots, C_m\}$, that is, every vertex in V is a member of exactly one of the sets C_0, \dots, C_m . Cluster C_0 is a singleton consisting of the depot vertex. For each $i \in V$, $\alpha(i)$ denotes the index of the cluster that vertex i belongs to. In other words, $i \in C_{\alpha(i)}$ with $\alpha(i) \in M = \{0, 1, 2, \dots, m\}$. E denotes the set of edges in graph G , where edges are defined from one cluster to the other only, i.e., inter-cluster edges do not exist. We therefore have $E = \{\{i, j\} | i, j \in V, \alpha(i) \neq \alpha(j)\}$. In several of the formulations we will be working with a directed graph. In the directed graph, we have an arc set A instead of the edge set E . For each edge $\{i, j\} \in E$ there exist two directed arcs (i, j) and (j, i) in A . Each cluster C_k with $k \geq 1$ has a nonnegative demand denoted by q_k , with $q_k > 0$ for $k = 1, \dots, m$ and $q_0 = 0$. There are K vehicles located at the depot with a

common capacity Q . The traversal of each edge $e : \{i, j\} \in E$ induces a traveling cost denoted by c_e (in case of a directed graph c_{ij} and c_{ji} may be different, but in this paper we restrict ourselves to symmetric instances).

The GVRP consists of finding a set of tours that all start and end at the depot for each of the K vehicles, such that *exactly* one vertex from each of the clusters is visited exactly once by any of the vehicles and the total demand served within each tour does not exceed the vehicle capacity Q , with an objective of minimizing the total cost of all the tours.

It is worth noting that penalties or bonuses can be applied to visiting certain vertices within a cluster. If we want to penalize vertex i with penalty p_i we simply add $0.5p_i$ to the cost of each edge/arc adjacent to i . In this way we can specify a preference for visiting certain vertices. This could, for example, be used in the first application mentioned in Section 1 to model that the cost of docking differs from harbor to harbor.

This section presents four formulations for the GVRP. All the formulations proposed here are based on two-index variables where a variable is defined for every arc (edge, for the undirected case). Our particular choice of such formulations is due to their success over three-index formulations where a variable is defined for every arc-vehicle combination (Letchford and Salazar-Gonzalez 2006). The first two of these formulations are based on the flow of a single-commodity and are polynomial in size. The latter two are directed and undirected formulations, and are both based on an exponential number of constraints.

Additional notation that will be used for these formulations is as follows: for any set S , $\delta(S) = \{(i, j) \in A | i \in S, j \notin S \text{ or } i \notin S, j \in S\}$, $A(S) = \{(i, j) \in A | i, j \in S\}$, $x(F) = \sum_{(i, j) \in F} x_{ij}$, $\delta^+(S) = \{(i, j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i, j) \in A | i \notin S, j \in S\}$. For simplicity, when $S = \{i\}$, we will write $\delta(i)$ as opposed to $\delta(\{i\})$. In the case of an undirected graph, the same notation holds for $\delta(S)$ and $x(F)$ except that pairs of arcs (i, j) and (j, i) are replaced by a single edge $\{i, j\}$ and we define $E(S) = \{\{i, j\} \in E | i, j \in S\}$.

As we present each formulation in the subsequent sections, we will also show how the strength of their linear programming relaxation compares to that of its predecessors. For any given formulation F , let F^L denote its linear programming relaxation obtained by allowing the integer variables to take continuous values within the lower and upper integer bounds, let $v(F)$ denote the value of its optimal solution and $c(F)$ denote the convex hull of its feasible solutions. We will denote the (infinite) set of instances of the GVRP with a symmetric cost matrix as \mathcal{GVRP} . We restrict ourselves to instances with symmetric cost matrices as these are the ones that all four formulations are able to handle. For an instance $I \in \mathcal{GVRP}$ we denote by $F(I)$ the concrete mathematical model

that arises from applying the formulation F on the instance I . As an example, for a formulation F the expression $c(F^L(I))$ denotes the set of solutions to the linear relaxation of F when applied to the instance I .

2.1. A Single-Commodity Formulation

This section presents a single-commodity flow formulation for the GVRP. The formulation is derived from the single commodity CVRP formulation proposed by Gavish and Graves (1978). It is based on a directed graph and uses a binary variable x_{ij} defined for every $(i, j) \in A$, which equals 1 if arc (i, j) is traversed by a vehicle, and 0 otherwise. A continuous variable $f_{ij} \geq 0, \forall (i, j) \in A$ indicates the amount that the vehicle carries from vertex i to vertex j . The formulation is presented as follows:

$$(\mathcal{F}_1) \quad \text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1)$$

subject to

$$x(\delta^+(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \quad (2)$$

$$x(\delta^-(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \quad (3)$$

$$x(\delta^+(C_0)) = K \quad (4)$$

$$x(\delta^+(i)) = x(\delta^-(i)) \quad \forall i \in V \quad (5)$$

$$f(\delta^+(i)) - f(\delta^-(i)) = \frac{1}{2} q_{\alpha(i)} (x(\delta^-(i)) + x(\delta^+(i))) \quad \forall i \in V \setminus \{0\} \quad (6)$$

$$0 \leq f_{ij} \leq Q x_{ij} \quad \forall (i, j) \in A \quad (7)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A. \quad (8)$$

In the formulation above, (2) and (3) correspond to the assignment constraints for each cluster to be visited exactly once. Constraints (5) are used to model route continuity. Constraints (4) ensure that exactly K vehicles depart from the depot. Constraints (6) model the flow of the commodity through each vertex by linking the flow and assignment variables, and (7) impose bounds on the flow on each arc. Note that although constraints (3) are implied by (2) and (5), they are included in the formulation for the sake of completeness. Similar (implied) constraints also exist in the two following formulations.

By taking the demands at clusters at the endpoints of the arc (i, j) into account, we note that the bounds on f_{ij} in inequalities (7) can be strengthened as follows:

$$q_{\alpha(i)} \leq f_{ij} \leq (Q - q_{\alpha(j)}) x_{ij} \quad \forall (i, j) \in A. \quad (9)$$

In the ensuing exposition, formulation \mathcal{F}_1 will be used with constraints (7) replaced by (9).

2.2. A Compact Single-Commodity Formulation

In this section, we present a simplified version of formulation \mathcal{F}_1 where the number of flow variables is reduced based on the idea that these only need to be defined for every pair of clusters as opposed to every pair of vertices. Hence, variables $f_{rs} \geq 0$, $r, s \in M$ now represent the amount that the vehicle carries from cluster C_r to C_s . The formulation is given as follows:

$$(\mathcal{F}_2) \quad \text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (10)$$

subject to

$$x(\delta^+(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \quad (11)$$

$$x(\delta^-(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \quad (12)$$

$$x(\delta^+(C_0)) = K \quad (13)$$

$$x(\delta^+(i)) = x(\delta^-(i)) \quad \forall i \in V \quad (14)$$

$$\sum_{s \in M \setminus \{r\}} f_{rs} = \sum_{p \in M \setminus \{r\}} f_{pr} + q_r \quad \forall r \in M \setminus \{0\} \quad (15)$$

$$0 \leq f_{rs} \leq Qx(C_r : C_s) \quad \forall r, s \in M, r \neq s \quad (16)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A.$$

In this formulation, constraints (11)–(14) have the same meaning as in formulation \mathcal{F}_1 , constraints (15) are used to model the increasing flow as the vehicle traverses through the tour, and (16) impose bounds on the flow on each inter-cluster arc. All other constraints are as explained in the preceding section. Constraints (16) can be strengthened in the same manner as were inequalities (7), as shown below.

$$q_r \leq f_{rs} \leq (Q - q_s)x(C_r : C_s) \quad \forall r, s \in M. \quad (17)$$

Formulation \mathcal{F}_2 will henceforth be used with constraints (16) replaced by (17).

The following proposition compares the linear programming bounds of \mathcal{F}_1 and \mathcal{F}_2 .

PROPOSITION 1. $v(\mathcal{F}_1^L(I)) \geq v(\mathcal{F}_2^L(I))$, $\forall I \in \mathcal{GVRP}$.

Proof We prove this by showing that given an instance I and a solution $(x^*, f^*) \in c(\mathcal{F}_1^L(I))$ we can always construct a solution $(\tilde{x}, \tilde{f}) \in c(\mathcal{F}_2^L(I))$ with the same cost as (x^*, f^*) . We claim that $\tilde{x}_{ij} = x_{ij}^*$, $\forall (i, j) \in A$, $\tilde{f}_{rs} = \sum_{i \in C_r} \sum_{j \in C_s} f_{ij}^*$, $\forall r, s \in M, r \neq s$ is such a solution. Obviously (\tilde{x}, \tilde{f}) has the same cost as (x^*, f^*) so we only need to show that $(\tilde{x}, \tilde{f}) \in c(\mathcal{F}_2^L(I))$. Equalities (11)–(14) are clearly satisfied, while (15) needs some more consideration. We have that

$$\sum_{s \in M \setminus \{r\}} \tilde{f}_{rs} - \sum_{p \in M \setminus \{r\}} \tilde{f}_{pr} = \sum_{s \in M \setminus \{r\}} \sum_{i \in C_r} \sum_{j \in C_s} f_{ij}^* - \sum_{p \in M \setminus \{r\}} \sum_{i \in C_p} \sum_{j \in C_r} f_{ij}^*$$

$$\begin{aligned}
 &= \sum_{i \in C_r} ((f^*(\delta^+(i)) - f^*(\delta^-(i)))) = \sum_{i \in C_r} \frac{1}{2} q_{\alpha(i)} (x^*(\delta^-(i)) + x^*(\delta^+(i))) \\
 &= \frac{1}{2} q_r (x^*(\delta^-(C_r)) + x^*(\delta^+(C_r))) = q_r,
 \end{aligned}$$

for all $r \in M \setminus \{0\}$, which is equivalent to (15). To see that (16) is satisfied we notice that

$$\tilde{f}_{rs} = \sum_{i \in C_r} \sum_{j \in C_s} f_{ij}^* \leq \sum_{i \in C_r} \sum_{j \in C_s} Q x_{ij}^* = Q x^*(C_r : C_s),$$

and that $\tilde{f}_{rs} \geq 0$ for all $r, s \in M, r \neq s$. \square

The proposition shows that the lower bound obtained by \mathcal{F}_1^L is at least as good as the one obtained by \mathcal{F}_2^L . The computational experiments in Section 6.2 show many examples where $v(\mathcal{F}_1^L)$ is strictly better than $v(\mathcal{F}_2^L)$.

Even though \mathcal{F}_2 is weaker than \mathcal{F}_1 it has its merits. Because it uses fewer variables and constraints its LP relaxation is easier to solve and therefore a branch-and-cut method based on \mathcal{F}_2 is able to process more branch-and-cut nodes per time unit than one based on the \mathcal{F}_1 formulation. The computational tests in Section 6 compares the performance of the two formulations in practice.

2.3. A Directed Formulation with an Exponential Number of Constraints

The preceding directed formulations use two sets of variables, one so-called “natural” (i.e., x variables) and the other called “auxiliary” (i.e., f variables), where the second set helps in enforcing special restrictions such as capacity and route continuity. We now present a formulation that is constructed using only the natural variables. Though the number of variables is reduced, this formulation requires an exponential set of constraints to model the special restrictions. The formulation, denoted by \mathcal{F}_3 is presented as follows:

$$(\mathcal{F}_3) \quad \text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{18}$$

subject to

$$x(\delta^+(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \tag{19}$$

$$x(\delta^-(C_k)) = 1 \quad \forall k \in M \setminus \{0\} \tag{20}$$

$$x(\delta^+(C_0)) = K \tag{21}$$

$$x(\delta^+(i)) = x(\delta^-(i)) \quad \forall i \in V \tag{22}$$

$$\begin{aligned}
 x(\delta^+(S)) &\geq \left\lceil \frac{q(S)}{Q} \right\rceil & S &= \bigcup_{k \in M'} C_k, \forall M' \subseteq M \setminus \{0\} \\
 x_{ij} &\in \{0, 1\} & & \forall (i, j) \in A.
 \end{aligned} \tag{23}$$

The term $q(S) = \sum_{i \in M | C_i \subseteq S} q_i$ is used to denote the total demand in set S . The purpose of including constraints (23) is two-fold. First, they ensure that no subtours will be formed among the customer vertices. Second, they eliminate any tours having a total demand greater than the vehicle capacity. We will henceforth refer to these constraints as *capacity constraints*.

Unfortunately, we are not able to state a comparison result between $v(\mathcal{F}_2^L)$ and $v(\mathcal{F}_3^L)$. This follows from the fact that the projection of inequalities (15) and (16) onto the x space for the unit-demand CVRP results in certain multistar inequalities (Gouveia 1995) and no specific dominance relation exists between these inequalities and the capacity constraints of the CVRP (Letchford and Salazar-Gonzalez 2006). The same result therefore holds between formulations \mathcal{F}_2^L and \mathcal{F}_3^L at least for a special case of the problem where $|C_k| = q_k = 1$ for all $k \in M$, and extends, under the light of Proposition 1, between formulations \mathcal{F}_1^L and \mathcal{F}_3^L .

2.4. An Undirected Formulation with an Exponential Number of Constraints

The last formulation we present is defined on an undirected graph and for this purpose it uses integer variables $z_e, e \in E$, which count the number of times the edge e is used. Only edges adjacent to the depot are allowed to be used more than once. These edges can be used twice. Occasionally we need to specify the endpoints of the edge e defining z_e . In that case we write z_{ij} where $e = \{i, j\}$ with the convention that $i < j$. The formulation is given as follows:

$$(\mathcal{F}_4) \quad \text{Minimize } \sum_{e \in E} c_e z_e \quad (24)$$

subject to

$$z(\delta(C_k)) = 2 \quad \forall k \in M \setminus \{0\} \quad (25)$$

$$z(\delta(C_0)) = 2K \quad (26)$$

$$z(\delta(S)) + 2 \sum_{(i,j) \in L: i \notin S} z(\{i\} : C_j) \leq 2 \quad \forall k \in M \setminus \{0\}, \forall S \subseteq C_k, \forall L \in \bar{L}_k \quad (27)$$

$$z(\delta(S)) \geq 2 \left\lceil \frac{q(S)}{Q} \right\rceil \quad S = \bigcup_{k \in M'} C_k, \forall M' \subseteq M \setminus \{0\} \quad (28)$$

$$z_e \in \{0, 1, 2\} \quad \forall e \in \delta(0) \quad (29)$$

$$z_e \in \{0, 1\} \quad \forall e \in E \setminus \delta(0). \quad (30)$$

In this formulation, constraints (25) ensure that each cluster is visited exactly once. Constraints (26) imply that K vehicles will leave the depot. Constraints (27) ensure that when a vehicle arrives to a certain vertex in a cluster, it departs from the same vertex. We will henceforth refer to these constraints as *same-vertex inequalities*. In this constraint, $\bar{L}_k = \{L : L \subseteq \bigcup_{i \in C_k} L_i, |L \cap L_i| = 1, \forall i \in$

C_k where $L_i = \{i\} \times (M \setminus \{0, \alpha(i)\})$, defined for all $i \in V \setminus \{0\}$. In other words, a set L_i consists of $|M| - 2$ two-tuples, the first element of which is always the vertex i and the second element of which is a cluster other than the depot cluster and $\alpha(i)$. The set \bar{L}_k consists of sets that are subsets of the union of all $L_i, i \in C_k$, and intersect with a single element of each set $L_i, i \in C_k$. Therefore, any member of the set \bar{L}_k contains a single tuple with the first component equal to any $i \in C_k$, and the second component is a cluster other than 0 and $\alpha(i)$. This definition ensures that in each same-vertex inequality, every vertex $i \in C_k$ for a given $k \in M \setminus \{0\}$ is mapped to exactly one cluster $j \in (M \setminus \{0, \alpha(i)\})$, and that these inequalities are written down for all pairs of possible mappings. Constraints (28) are the undirected version of the capacity constraints (23).

To see how these the same-vertex inequalities work, consider Figure 1. In the example shown in the figure, two different vertices (i and j) are used to enter and leave cluster C_k . If $p \neq 0$ and $q \neq 0$ then we can eliminate such a solution by, for example, setting $S = C_k \setminus \{i, j\}$ and mapping vertices i and j to clusters $C_{\alpha(p)}$ and $C_{\alpha(q)}$, respectively. This results in (27) having a left hand side value of 4. If for example $p = 0$ then the solution can be eliminated by setting $S = C_k \setminus \{j\}$ and mapping vertex j to cluster $C_{\alpha(q)}$, which results in (27) having a left hand side value of 3.

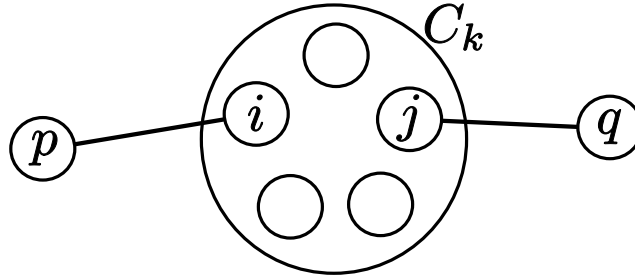


Figure 1 An example of an integral feasible solution in the absence of a same-vertex inequality (27).

The following proposition proves the validity of the same-vertex inequalities.

PROPOSITION 2. *The same-vertex inequalities (27) are valid for the GVRP.*

Proof Consider an inequality (27) given by the selection of $k \in M \setminus \{0\}$, $S \subseteq C_k$ and $L \in \bar{L}_k$. Any feasible GVRP solution satisfies either $z(\delta(S)) = 0$ or $\sum_{i \in C_k \setminus S} z(\{i\} : C_j) = 0$ as only one vertex from the cluster k can be visited. The inequality is clearly valid when $\sum_{i \in C_k \setminus S} z(\{i\} : C_j) = 0$ as $z(\delta(S)) \leq 2$ due to (25). If $z(\delta(S)) = 0$ the inequality is still valid. To see this, first note that $z(\{i\} : C_j) \leq 1$ for all $i \in \{1, \dots, n\}$ and $j \in M \setminus \{0, \alpha(i)\}$. Second, note that for every cluster C_k , $z(\delta(i)) = 0$ for all but one vertex $i \in C_k$. \square

The only situation that the same-vertex inequalities do not handle is when $z_{0i} = 1$ and $z_{0j} = 1$ for $i, j \in C_k, i \neq j$ for some $k \in M \setminus \{0\}$. Such a solution is obviously infeasible. Note that this case is a convex combination of two feasible solutions in which vertices i and j are visited by a vehicle that comes from the depot immediately returns to the depot. Hence, it is not possible to separate this solution by a valid inequality. However, this case will not occur in an optimal solution unless the costs of the edges $\{0, i\}$ and $\{0, j\}$ are exactly the same. In such a case the output can be corrected with a simple post-processing that converts the solution to one in which a vehicle visits only one of the vertices and goes back to the depot. Note that identifying such a solution takes $O(|V|^2)$ time, simply by computing the degree of every vertex and identifying the vertices with degree 1.

The following proposition compares of the strengths \mathcal{F}_3^L and \mathcal{F}_4^L in terms of the bounds provided by their linear programming relaxations.

PROPOSITION 3. $v(\mathcal{F}_4^L(I)) \geq v(\mathcal{F}_3^L(I)), \forall I \in \mathcal{GVRP}$.

Proof Similar to the proof of Proposition 1 we show that given an instance I and a solution $z^* \in c(\mathcal{F}_4^L(I))$ we can always construct a solution $x^* \in c(\mathcal{F}_3^L(I))$ with the same cost as z^* . We claim that

$$x_{ij}^* = \begin{cases} \frac{1}{2}z_{ij}^* & \forall (i, j) \in A, i < j \\ \frac{1}{2}z_{ji}^* & \forall (i, j) \in A, i > j, \end{cases}$$

is such a solution. The cost of x^* is obviously the same as that of z^* so it remains to show that $x^* \in c(\mathcal{F}_3^L(I))$. This amounts to showing that x^* satisfies (19)–(23) which is straightforward. For example, we have that $x^*(\delta^+(C_k)) = \sum_{i \in C_k} \sum_{j \in V \setminus C_k} x_{ij}^* = \frac{1}{2}z^*(\delta(C_k)) = 1$ for all $k \in M \setminus \{0\}$ which shows that x^* satisfies (19) and the other constraints follow in the same way. \square

The computational experiments in Section 6.2 show many examples where $v(\mathcal{F}_4^L)$ is strictly better than $v(\mathcal{F}_3^L)$.

2.4.1. Valid Inequalities from the CVRP. It is clear that the GVRP and CVRP are closely related problems. In this section we show how valid inequalities from the 2-index formulation of the CVRP can be used to strengthen the linear relaxation of \mathcal{F}_4 . Like the GVRP, the CVRP can be defined on an undirected graph $G' = (V', E')$, where $V' = \{0, \dots, n'\}$. Each vertex $i' \in V'$ has an associated demand $q'_i \geq 0$ with $q'_0 = 0$ and the capacity of the of the K' identical vehicles is denoted Q' . The standard IP-model for the CVRP is as follows:

$$(\mathcal{F}_C) \quad \text{Minimize} \quad \sum_{e \in E'} c'_e y_e \quad (31)$$

subject to

$$y(\delta(k)) = 2 \quad \forall k \in \{1, \dots, n'\} \quad (32)$$

$$y(\delta(0)) = 2K' \quad (33)$$

$$y(\delta(S)) \geq 2 \left\lceil \frac{q'(S)}{Q'} \right\rceil \quad \forall S \subseteq \{1, \dots, n'\}, |S| \geq 2 \quad (34)$$

$$y_e \in \{0, 1, 2\} \quad \forall e \in \delta(0) \quad (35)$$

$$y_e \in \{0, 1\} \quad \forall e \in E' \setminus \delta(0), \quad (36)$$

where y variables are defined in a similar manner to the z variables. Every GVRP instance induces a CVRP instance by shrinking the vertices in each cluster to a single vertex. More formally, the CVRP instance obtained from a GVRP instance has the following characteristics: $K' = K, Q' = Q, n' = m, V' = \{0, \dots, n'\}, E' = \{\{k, l\} : k, l \in V', \exists i \in C_k, j \in C_l \text{ such that } \{i, j\} \in E\}, q'_k = q_k \forall k \in \{0, \dots, n'\}$. The definition of c'_e is unimportant because we are concerned with feasibility in this section. A feasible solution to a GVRP instance can be turned into a feasible solution to the induced CVRP instance as the following Lemma shows.

LEMMA 1. *A solution y^* , induced by a solution $z^* \in c(\mathcal{F}_4)$ through the relation $y_{kl}^* = z^*(C_k : C_l) \forall \{k, l\} \in E'$ is always such that $y^* \in c(\mathcal{F}_C)$.*

Proof We show that y_{kl}^* is feasible for (32)–(36). This amounts to showing that each constraint is satisfied. For (32)–(34) we do this by substitution. For example, substituting on the left hand side of (32) gives

$$y^*(\delta(k)) = \sum_{l \in V' \setminus \{k\}} y_{kl}^* = \sum_{l \in M \setminus \{k\}} z^*(C_k : C_l) = z^*(\delta(C_k)) = 2, \forall k \in \{1, \dots, k\},$$

and substituting on the left hand side of (34) yields,

$$y^*(\delta(S')) = \sum_{k \in S'} \sum_{l \in M \setminus S'} z^*(C_k : C_l) = z^*(\delta(S')) \geq 2 \left\lceil \frac{q(S)}{Q} \right\rceil = 2 \left\lceil \frac{q'(S')}{Q'} \right\rceil,$$

where $S = \cup_{k \in S'} C_k$ and the inequality holds for all $S' \subseteq \{1, \dots, n'\}, |S'| \geq 2$. To see that (35) and (36) are satisfied we first notice that each element in y^* is a non-negative integer because it is a sum of non-negative integers. We just need to show that $y_e^* \leq 2$ when $e \in \delta(0)$ and $y_e^* \leq 1$ otherwise. For any edge $e = \{0, l\} \in \delta(0)$ we have $y_e^* \leq y(\delta(l)) = 2$. For an edge $e \in V' \setminus \delta(0)$ assume that $y_e^* \geq 2$. In that case we must have $y_e^* = 2$ due to (32), that is, $z^*(C_k : C_l) = 2$, for $e = \{k, l\}, k \neq l, k \neq 0, l \neq 0$. This, together with (25), implies that $z^*(\delta(C_k \cup C_l)) = 0$ which means that (28) is violated. This is in contradiction with z^* being a feasible GVRP solution and therefore $y_e^* \leq 1, \forall e \in V' \setminus \delta(0)$. \square

The following corollary follows directly from Lemma 1.

COROLLARY 1. *If $\sum_{k \in V'} \sum_{l \in V', k < l} a_{kl} y_{kl} \leq b$ is a valid inequality for the induced CVRP instance then $\sum_{k \in M} \sum_{l \in M, k < l} a_{kl} z(C_k : C_l) \leq b$ is a valid inequality for the original GVRP instance.*

The capacity inequalities (28) are an example of valid inequalities that stem from valid inequalities on the induced CVRP instance. The capacity inequalities for the CVRP can be written:

$$-y(\delta(S')) \leq -2 \left\lceil \frac{q'(S')}{Q'} \right\rceil \quad \forall S' \subseteq \{1, \dots, n'\}, |S'| \geq 2,$$

and using Corollary 1 we get that

$$-z(\delta(S)) = - \sum_{\{k,l\} \in \delta(S')} z(C_k : C_l) = -y(\delta(S')) \leq -2 \left\lceil \frac{q'(S')}{Q'} \right\rceil = -2 \left\lceil \frac{q(S)}{Q} \right\rceil, \\ \forall S' \subseteq \{1, \dots, n'\}, |S'| \geq 2, S = \cup_{k \in S'} C_k,$$

is a valid inequality for the GVRP. This inequality is equivalent to (28).

If a separation algorithm for the CVRP inequality is available, then this algorithm can be used to separate the induced GVRP inequality as well: a fractional solution $z^* \in c(\mathcal{F}_4^L)$ can be transformed into a fractional solution $y^* \in c(\mathcal{F}_C^L)$ by using the transformation from Lemma 1. The CVRP separation algorithm is run with y^* as input and if a violated inequality is detected then this inequality is transformed into an inequality for the GVRP using Corollary 1, which would imply that the resulting GVRP inequality is violated by z^* .

We should mention that any valid inequality for \mathcal{F}_4 is easily transformed into a valid inequality for \mathcal{F}_1 – \mathcal{F}_3 by performing the substitution $z_{ij} = x_{ij} + x_{ji} \forall i, j \in V, i < j$. This shows that the valid inequalities obtained from the CVRP also can be used to strengthen the three previous formulations of the GVRP. In order to keep the computational comparisons of the four formulations as clean and simple as possible we have not performed experiments with valid inequalities induced from the CVRP polytope apart from the capacity constraints that already are part of formulations \mathcal{F}_3 and \mathcal{F}_4 .

3. Branch-and-Cut Algorithms

In this section we describe the branch-and-cut algorithms devised to solve formulations \mathcal{F}_3 and \mathcal{F}_4 . The algorithms are implemented using CPLEX 10.0 and the Concert framework. To simplify the description we only describe the branch-and-cut algorithm for \mathcal{F}_4 . The implementation of the branch-and-cut algorithm for \mathcal{F}_3 is similar. The initial relaxation contains (24)–(26) while (27) and (28) are identified and added dynamically using the separation algorithms we describe in Sections 3.1 and 3.2. CPLEX's own branching scheme (branches on a single variable in the formulation) is employed and CPLEX's implementation of strong branching is enabled. The branch-and-cut nodes are processed using a best-bound strategy.

3.1. Separation of Same-Vertex Inequalities (27)

To identify the violated members of the constraint set (27), we use an algorithm which consists of two phases. In the first phase we calculate and record the values of $z(\delta(i))$, $\max_{l \in M: l \neq k} \{z(\{i\} : C_l)\}$, and $\arg \max_{l \in M: l \neq k} \{z(\{i\} : C_l)\}$ for every vertex $i \in C_k$ and each cluster $k \neq 0$. This phase takes $O(n^2)$ time. In the second phase, we analyze each cluster C_k and every vertex $i \in C_k$. If $z(\delta(i)) \geq 2 \max_{l \in M: l \neq k} \{z(\{i\} : C_l)\}$, then we insert i into the set S . Else, we add the pair $(i, \arg \max_{l \in M: l \neq k} \{z(\{i\} : C_l)\})$ to the set $L \subset \bar{L}_k$. This procedure simply maximizes the left hand side of (27). If the final value of the left hand side is greater than 2, then we add the violated inequality to the cut pool. The complexity of the second phase is $O(n)$, and the overall complexity of the separation procedure is $O(n^2)$.

3.2. Separation of Capacity Constraints (28)

The capacity constraints are separated using the method outlined in Section 2.4.1. In order to separate the capacity inequalities for the CVRP we use the heuristic routines made available by Lysgaard (2003).

4. Preprocessing

We now give a simple yet effective preprocessing algorithm for the GVRP that is able to reduce the size of some instances by removing *dominated* vertices. A vertex $i \in V \setminus \{0\}$ is said to be *dominated* if 1) for all $p, q \in V \setminus C_{\alpha(i)}$, $\alpha(p) \neq \alpha(q)$, $q_{\alpha(i)} + q_{\alpha(p)} + q_{\alpha(q)} \leq Q$ there exists a vertex $j \in C_{\alpha(i)}$, $j \neq i$ such that $c_{pi} + c_{iq} \geq c_{pj} + c_{jq}$ and 2) there exists a vertex $j \in C_{\alpha(i)}$, $j \neq i$ such that $c_{0i} \geq c_{0j}$.

PROPOSITION 4. *Removing a single dominated vertex from a GVRP instance does not change the value of the optimal solution.*

Proof Let $i \in V \setminus \{0\}$ be a dominated vertex. If i is not visited by the optimal solution then it obviously does not change the value of the optimal solution to remove i from the instance. Assume now that i is visited by the optimal solution. If i is visited by a route visiting exactly one customer then it is possible to exchange i with another vertex from $C_{\alpha(i)}$ without increasing the cost of the solution. This follows from 2) in the definition of dominated vertices. If i is visited on a longer route then it is surrounded by vertices $p \in V$ and $q \in V$ where $q_{\alpha(i)} + q_{\alpha(p)} + q_{\alpha(q)} \leq Q$ and either $p \neq 0$ or $q \neq 0$. In this case p and q fulfill the requirement of condition 1) in the definition and i can be exchanged with another vertex from $C_{\alpha(i)}$ without increasing the cost of the solution. \square

The preprocessing algorithm simply tests if each vertex is dominated. If so, it is removed from the instance. After removing a vertex the next vertices are tested for dominance relative to the reduced instance. It is not valid to first test all vertices for dominance and then remove the dominated

ones. The worst time complexity of this procedure is $O(n^5)$. This might seem extremely slow, but in practice (for the instance sizes considered), the algorithm is quite fast. Also notice that the time complexity is reduced to $O(n^4)$ if the maximum size of the clusters are bounded by a constant.

5. Large Neighborhood Search (LNS) Heuristic

Prior to the execution of the branch-and-cut algorithm, upper bounds are calculated using a simple large neighborhood search (LNS) metaheuristic which is a variation of that proposed earlier by Shaw (1998). The variant used here is similar to that proposed by Ropke and Pisinger (2006) which successfully has been applied to a number of vehicle routing problems in Pisinger and Ropke (2007). The heuristic improves an initial solution by repeatedly removing a set of vertices (destroying the solution) and reinserting them or *equivalent* vertices in the solution (repairing the solution). By equivalent we mean vertices from the clusters of the removed vertices. The number of vertices to remove is $m \cdot r$ where r is a random number in the interval $[0.1, 0.4]$. The number r differs from one iteration to the next. The vertices to remove are selected randomly. The vertices are reinserted using a *regret* heuristic. The regret heuristic calculates the cost of inserting each vertex from each unassigned cluster in each route. It chooses to insert a vertex from the cluster for which the difference in cost between the best and the second best route is the largest. The heuristic inserts the vertex (considering only the vertices from the selected cluster) that can be inserted most cheaply and it is inserted at its best possible position. One can say that the regret heuristic attempts to look ahead by considering what would happen if a cluster/vertex cannot be inserted where it fits best. The regret heuristic is described in further details in Ropke and Pisinger (2006). A randomized version of the regret heuristic is constructed by applying noise to the evaluation of insertion costs. The randomized regret heuristic is used with 50% probability and the ordinary regret heuristic is used otherwise.

The destroy-repair process is embedded in a simulated annealing framework (see Kirkpatrick et al. 1983). Applying the destroy-repair operation to a solution s results in a solution s' . The new solution s' is always accepted if $f(s') \leq f(s)$ and is accepted with probability

$$e^{-\frac{f(s)-f(s')}{T}},$$

otherwise. Here $f(s)$ be the cost of solution s , T is the temperature, initialized at 1500, and updated as $T = \alpha T$ at every iteration, with the constant α selected such that $T = 0.05$ after 25000 iterations at which point the LNS heuristic stops. The initial solution is constructed by generating K routes, each containing a single seed vertex and inserting the rest of the clusters/vertices by the regret

heuristic. The seed vertices are found by first selecting K clusters far from each other and from the depot and then selecting the vertex closest to the depot from each of these clusters. The LNS heuristic is allowed to visit solutions that do not serve all clusters. This is necessary because the initial solution not necessarily serves all clusters. A large penalty is applied per unserved cluster to discourage solutions with unserved clusters. All parameters for the LNS heuristic were selected after performing a few tests with the algorithm. A thorough parameter tuning process has not been carried out.

6. Computational Results

This section presents the results of comparison of the four formulations presented in Section 2, as well as those of the extensive computational experiments run with the dominant formulation. The section is organized in six subsections. The first describes how the instances are generated by performing suitable modifications to available literature instances. The second subsection presents the results obtained by comparing the lower bounds provided by all four formulations. In the third subsection, we compare the four formulations on a limited set of instances, where \mathcal{F}_3 and \mathcal{F}_4 are solved within a branch-and-cut framework. Finally, subsection four through six present the results obtained with the superior formulation on an extensive set of modified literature instances.

All the experiments of this section were performed using CPLEX 10.0. The computations were done on an AMD Opteron 250 computer (2.4 GHz).

6.1. Problem Instances

The experiments were conducted on three sets of problem instances. The instances in the first two sets are generated through an adaptation of the existing instances in the CVRP-library available at <http://branchandcut.org/VRP/data/>, in a similar manner to that of Fischetti et al. (1997) who have derived GTSP instances from the existing TSP instances. The first set is composed of medium-sized instances and derived using the A, B and P instances in the CVRP-library with the number of vertices being anywhere from 16 to 101. The second set consists of larger-sized instances, derived from the M and G instances in the CVRP-library, encompassing 101 to 262 vertices. Finally, the third set is a singleton containing the instance described in Ghiani and Improta (2000). We will now describe how the existing CVRP instances were translated into GVRP instances.

For a given CVRP instance with n vertices, the number of clusters is calculated as $m = \lceil n/\theta \rceil$ in the spirit of Fischetti et al. (1997) using $\theta = 2$ and $\theta = 3$. Fischetti et al. (1997) used $\theta = 5$ but doing so for the CVRP instances created GVRP instances that were too easy to solve. The clustering method chooses m seed-vertices by selecting vertices that are as far from one another

as possible. The assignment of each non-seed vertex to a cluster is performed using the CLUSTERING procedure of Fischetti et al. (1997). For each cluster, the demands are calculated by $q_k = \gamma \sum_{i \in V | k = \alpha(i)} \omega_i$, where ω_i refers to the vertex demands in the original CVRP instances and γ is used to scale the demand. In cases where q_k exceed the total available capacity of the vehicle fleet, excess demand was “cut-off”. As for the scaling factor, we first experimented with $\gamma = 1.0$ to keep the original demands. This particular choice resulted in rather short routes and many clusters with demand $q_k = Q$. Therefore we chose to use $\gamma = 1/m$ in the generation process so that the resulting instances are as similar in structure (e.g., length of routes) to that of the original CVRP instances as possible. The original number of vehicles for each instance was not adopted as for many instances the number would be too high as compared to the number of clusters. Instead, this number was calculated for each instance by solving an associated bin-packing problem using the implementation in Martello and Toth (1990).

The naming of the generated instances follows the general convention of the CVRP instances available online, albeit slightly modified to incorporate additional parameters. The naming follows the general format $X-nY-kZ-C\Omega-V\Phi$, where X corresponds to the type of the instance, Y refers to the number of vertices, Z corresponds to the number of vehicles in the original CVRP instance, Ω is the number of clusters and Φ is the number of vehicles in the GVRP instance. The entries in the cost matrix are calculated using Euclidean distances and are, unless otherwise stated, rounded to the nearest integer value. The instances are available at <http://www.personal.soton.ac.uk/tb12v07/gvrp.html>.

6.2. Lower Bound Comparisons

The first set of results pertain to the comparison of the lower bounds obtained by solving the LP relaxation of the formulations \mathcal{F}_1 - \mathcal{F}_4 . The formulations were tested on a limited set of converted A, B and P instances with $n \leq 45$. For this experiment, all processing routines and automatic addition of cuts (by CPLEX) have been turned off. The results of this experiment are given in Table 1. This table presents, for each formulation \mathcal{F}_i ($i = 1, \dots, 4$), the value of the lower bound obtained by this formulation ($v(\mathcal{F}_i)$).

The results presented in Table 1 are in line with the theoretical findings as stated in Propositions 1 and 3. We note that the bounds obtained using \mathcal{F}_3 are better as compared to those found by \mathcal{F}_1 for all of the set A and B instances, whereas there are quite a few cases where the opposite situation holds in set P instances. The results, however, indicate conclusively that the best results in terms of lower bounds are obtained with \mathcal{F}_4 .

Table 1 Comparison of the linear programming bounds of the four formulations on a limited set of instances

Instance	$v(\mathcal{F}_1)$	$v(\mathcal{F}_2)$	$v(\mathcal{F}_3)$	$v(\mathcal{F}_4)$
A-n45-k6-C23-V4	526.9710	480.8311	534.0000	576.5000
A-n45-k7-C23-V4	593.5303	549.9788	598.0000	623.3240
A-n46-k7-C23-V4	538.1986	500.9288	547.6000	560.5000
A-n48-k7-C24-V4	566.6433	526.8580	582.5000	616.0000
A-n53-k7-C27-V4	530.4432	492.8231	542.3330	578.7870
A-n54-k7-C27-V4	577.2945	540.5200	636.6670	653.9750
A-n55-k9-C28-V5	606.5451	562.8113	638.0000	657.5280
A-n60-k9-C30-V5	678.0961	635.3473	712.0000	739.5830
A-n61-k9-C31-V5	563.5525	538.2837	594.7140	615.0710
A-n62-k8-C31-V4	637.2069	602.0016	679.5000	712.8330
A-n63-k10-C32-V5	700.8570	654.2022	732.8120	753.0450
A-n63-k9-C32-V5	801.9550	754.9002	836.1600	859.5000
A-n64-k9-C32-V5	696.6793	656.6579	700.3750	725.5830
A-n65-k9-C33-V5	597.7375	579.3059	632.4670	661.0000
A-n69-k9-C35-V5	596.6550	570.8476	611.6670	640.8900
A-n80-k10-C40-V5	873.9827	806.5891	887.6670	908.4420
B-n50-k7-C25-V4	382.8571	378.1883	436.0000	440.6670
B-n50-k8-C25-V5	763.1153	749.4290	877.6110	880.5500
B-n51-k7-C26-V4	534.2611	525.6736	632.5000	645.7500
B-n52-k7-C26-V4	338.5499	331.4444	439.0000	446.5000
B-n56-k7-C28-V4	335.8497	329.0657	470.0000	479.3060
B-n57-k7-C29-V4	567.3769	556.8076	734.0000	741.5000
B-n57-k9-C29-V5	843.0059	832.9838	923.0000	928.1560
B-n63-k10-C32-V5	728.3088	713.5110	780.0000	795.5000
B-n64-k9-C32-V5	397.2110	387.5947	487.0000	503.5000
B-n66-k9-C33-V5	697.5809	693.0690	788.6670	799.9230
B-n67-k10-C34-V5	550.5922	539.9730	651.0000	660.1670
B-n68-k9-C34-V5	606.6277	596.5802	689.0000	695.7500
B-n78-k10-C39-V5	677.9088	659.8112	773.3130	787.2010
P-n45-k5-C23-V3	289.8923	272.0348	304.0000	325.4380
P-n50-k10-C25-V5	367.9826	342.9248	360.5000	371.5000
P-n50-k7-C25-V4	309.6226	290.8728	308.0000	324.5000
P-n50-k8-C25-V4	333.6222	308.6161	326.2860	337.5410
P-n51-k10-C26-V6	401.1189	363.2288	382.5000	400.5360
P-n55-k10-C28-V5	369.3851	344.6828	365.6670	377.8750
P-n55-k15-C28-V8	497.2703	473.8719	494.3330	504.0010
P-n55-k7-C28-V4	312.2467	294.7346	316.5000	336.2500
P-n55-k8-C28-V4	317.9056	298.7594	323.4290	337.5000
P-n60-k10-C30-V5	389.2288	367.4788	391.2500	401.5890
P-n60-k15-C30-V8	507.9790	482.3177	505.2140	517.0500
P-n65-k10-C33-V5	446.7878	415.2066	437.0000	452.5830
P-n70-k10-C35-V5	445.9197	421.7578	444.1670	462.9760
P-n76-k4-C38-V2	340.2125	328.2294	342.2500	370.0620
P-n76-k5-C38-V3	357.3376	343.4320	362.0000	390.6480
P-n101-k4-C51-V2	384.3812	375.3025	407.2500	438.5420

6.3. Overall Comparisons

Although the lower bound results presented in Table 1 already imply that formulations \mathcal{F}_3 and \mathcal{F}_4 are likely to be more efficient than the remaining two for the solution of the GVRP, we have performed additional experiments that compare the four formulations with respect to their relative efficiency. For purposes of comparison the four formulations have been tested only on a limited set of instances with $n \leq 45$. Each instance has been solved using formulations \mathcal{F}_1 and \mathcal{F}_2 using CPLEX’s own branch-and-cut algorithm, whereas formulations \mathcal{F}_3 and \mathcal{F}_4 have been solved using a branch-and-cut algorithm described in Section 3. A computational time limit of two hours has been imposed on all four formulations for comparison purposes. The results of this experiment are

given in Table 2. The first column presents the names of the instances, the second column titled n_p shows the number of vertices remaining after the preprocessing routine (see Section 4) has been run on the instance and the third column titled **Opt** presents the value of the optimal solution for each instance. In the remaining columns, $t(\mathcal{F})$ shows the time required (in CPU seconds) to solve each instance to optimality and $bb(\mathcal{F})$ is the number of vertices in the branch and bound tree for each formulation \mathcal{F} .

Table 2 Comparison of the performance of four formulations on solving a limited set of instances

Instance	n_p	$t(\mathcal{F}_1)$	$bb(\mathcal{F}_1)$	$t(\mathcal{F}_2)$	$bb(\mathcal{F}_2)$	$t(\mathcal{F}_3)$	$bb(\mathcal{F}_3)$	$t(\mathcal{F}_4)$	$bb(\mathcal{F}_4)$
A-n32-k5-C16-V2	30	7200.27 [§]	64809	7200.27 [†]	150827	1436.22	6358	113.15	1847
A-n33-k5-C17-V3	32	7190.92	75408	2156.97	67941	5.39	17	1.64	38
A-n33-k6-C17-V3	29	76.74	1589	112.13	4221	3.16	6	0.73	5
A-n34-k5-C17-V3	30	239.97	3509	66.25	2387	3.46	7	0.84	5
A-n36-k5-C18-V2	36	7200.34 [§]	43612	7200.33 [§]	162547	99.40	1117	31.48	612
A-n37-k5-C19-V3	34	25.06	310	42.29	1075	4.04	4	0.76	1
A-n37-k6-C19-V3	36	7200.34 [§]	47809	7200.33 [§]	139903	65.65	238	28.20	264
A-n38-k5-C19-V3	37	1238.37	12778	2893.75	79687	2.83	0	2.96	27
A-n39-k5-C20-V3	36	7200.38 [§]	46587	7200.39 [§]	103732	40.72	280	45.57	544
A-n39-k6-C20-V3	36	7200.38 [§]	46330	7200.39 [†]	92471	26.69	168	4.88	42
A-n44-k6-C22-V3	41	7200.38 [†]	38794	7200.38 [†]	87213	140.67	336	23.22	210
A-n45-k6-C23-V4	39	7200.47 [§]	42063	7200.45 [§]	80144	7.46	44	6.79	112
A-n45-k7-C23-V4	41	7200.46 [§]	29696	7200.45 [§]	77668	7082.46	8907	1465.19	3184
B-n31-k5-C16-V3	27	49.32	743	17.80	574	0.30	0	0.13	0
B-n34-k5-C17-V3	29	7200.31 [§]	79260	7200.31 [†]	241982	1.74	0	0.08	0
B-n35-k5-C18-V3	32	5089.85	74283	7200.36 [†]	169215	0.42	0	0.08	0
B-n38-k6-C19-V3	35	4164.30	43439	853.03	30302	3.17	11	0.69	3
B-n39-k5-C20-V3	35	7200.41 [§]	48977	5540.29	136599	0.55	0	0.20	2
B-n41-k6-C21-V3	34	7200.38 [§]	49520	7200.37 [§]	118099	18.44	144	2.62	79
B-n43-k6-C22-V3	40	7200.41 [§]	32396	7200.39 [§]	112163	27.24	153	9.22	82
B-n44-k7-C22-V4	39	7200.43 [§]	35124	7200.41 [§]	93556	11.94	16	3.26	17
B-n45-k5-C23-V3	43	7200.46 [§]	35615	7200.48 [†]	95429	2.02	0	0.60	5
B-n45-k6-C23-V4	39	7200.47 [§]	32839	7200.46 [§]	89396	463.59	3516	53.71	717
P-n16-k8-C8-V5	13	0.32	0	0.17	1	0.05	0	0.02	0
P-n19-k2-C10-V2	16	0.44	0	0.44	18	0.07	0	0.01	0
P-n20-k2-C10-V2	17	0.61	20	1.22	62	0.65	0	0.01	0
P-n21-k2-C11-V2	19	1.11	11	1.02	74	0.29	0	0.04	0
P-n22-k2-C11-V2	20	0.75	1	1.63	100	0.28	0	0.11	3
P-n22-k8-C11-V5	21	14.74	568	3.00	349	0.28	0	0.04	0
P-n23-k8-C12-V5	18	72.60	4157	26.97	2789	3.70	24	0.75	17
P-n40-k5-C20-V3	37	351.39	3704	717.17	14290	4.81	69	2.09	29
P-n45-k5-C23-V3	43	3024.82	21050	7200.47 [§]	85876	3.74	7	2.18	16

†: Optimality not verified within the two-hour time limit.

§: No feasible solution found within the two-hour time limit.

Results shown in Table 2 indicate that formulation \mathcal{F}_4 is the superior formulation among the four presented in this paper. Although similar computations can be performed on a larger set of instances for further comparison of the four formulations, we believe that the results given in Table 2 provide sufficient evidence for the supremacy of formulation \mathcal{F}_4 . We therefore tested only this formulation on other sets of problems in the next section. The results in Table 2 do not enable us recommend one of the two polynomially sized formulations \mathcal{F}_1 or \mathcal{F}_2 instead of the other.

6.4. Performance of the Branch-and-Cut Algorithm Based on \mathcal{F}_4

This section provides the results of an extensive set of experiments performed using \mathcal{F}_4 on a large test bed. In these experiments we allowed CPLEX to generate its own generic cuts (e.g. disjunctive or Gomory cuts), and a limit of two hours (7200 seconds) was imposed on the solution time. Before starting the branch-and-cut algorithm the instance was reduced by the preprocessing routine (Section 4) and the LNS heuristic was executed once to obtain an upper bound.

For the sake of brevity, we present here only a summary of the results in Table 3. Full results detailed for each instance can be found in the Appendix. The summary table presents, for each set of instances, the type of instance (Type), the value of θ in generating the instances (θ), the average number of vertices removed from the problem (\bar{n}'), the average solution time (\bar{T}) required by the branch-and-cut algorithm and the average time spent by the LNS heuristic for calculating a feasible solution for the problem (\bar{t}_{LNS}). The column titled g shows the percentage gap of optimality and is calculated as,

$$100 \frac{v_{IP} - v_{LP}}{v_{IP}}, \quad (37)$$

where v_{IP} is the value of either the best or the optimal solution obtained within the time limit, and v_{LP} is the value of the best lower bound after branching. In cases where no feasible solution was found, v_{IP} was replaced with the value of the best solution as output by the LNS heuristic.

In the subsequent columns, \overline{BB} denotes the average number of nodes in the branch and bound tree, \overline{CC} and \overline{SNC} present the average number of violated capacity inequalities (28) and same-vertex inequalities (27), respectively, and \bar{t}_{CC} and \bar{t}_{SNC} show the corresponding average separation time (in CPU seconds). Finally, the last column shows how many instances out of the total number of instances in each set were successfully solved to optimality within the given time limit.

Table 3 Summary of experiments using \mathcal{F}_4

Type	θ	\bar{n}'	\bar{T}	\bar{t}_{LNS}	g	\overline{BB}	\overline{CC}	\bar{t}_{CC}	\overline{SNC}	\bar{t}_{SNC}	
A	2	3.26	861.14	0.48	0.1984	1094.07	683.37	5.79	210.41	0.11	25/27
B	2	5.35	159.49	0.50	0.0000	400.17	308.43	1.60	100.61	0.03	23/23
P	2	2.08	1413.64	0.53	0.3933	1652.92	810.38	9.50	193.92	0.15	20/24
A	3	6.63	804.44	0.32	0.2539	1290.85	251.26	3.11	323.48	0.12	25/27
B	3	9.74	33.88	0.32	0.0000	270.30	93.52	0.28	124.57	0.02	23/23
P	3	4.42	712.87	0.34	0.0300	1772.46	221.75	3.30	281.25	0.25	23/24

Table 3 shows that the proposed branch-and-cut algorithm based on formulation \mathcal{F}_4 is highly successful in obtaining optimal solutions for a very high proportion of the set of instances generated and tested in this paper. The table also indicates that the average separation time for both the same-vertex inequalities (27) and capacity inequalities (28) are very low. Overall the algorithm can

be seen as successful for the solution of GVRP instances of similar dimensions as the ones tested in this paper.

In order to evaluate the effect of the preprocessing procedure and the LNS heuristic we have performed additional experiments on a limited set of instances with one or both of these features turned off. The limited set of instances consists of the A-dataset with both $\theta = 2$ and $\theta = 3$, this limited dataset contains 54 instances. We ran the branch-and-cut algorithm for formulation \mathcal{F}_4 with a time-limit of two hours. The standard configuration A with both features turned on solved 50 instances, configuration B without the LNS heuristic solved 48 instances while configuration C without the preprocessing procedure solved 49 instances. Configuration D with both features turned off solved 46 instances.

46 of the 54 instances were solved by all configurations. For these instances the average solution time was 113 seconds, 409 seconds, 131 seconds and 520 seconds for configurations A, B, C and D, respectively. We see that both the heuristic and the preprocessing routine have a positive impact on the performance of branch-and-cut algorithm and good upper bounds are especially important.

6.5. Results with Large-Scale Instances

This section presents the results of the computational experiments on the large-scale instances. Due to the size of these instances, the time limit imposed on the running time of the branch-and-cut algorithm has been increased to six hours (21600 CPU seconds). The columns of these tables are explained in the Appendix. All other settings of the algorithm are as described in the previous section. We give the results for instances generated using $\theta = 2$ and $\theta = 3$ in Tables 4 and 5, respectively.

Table 4 Computational results for instances generated using $\theta = 2$ and $\gamma = 1/m$

Instance	n'	T	t_{LNS}	v_{LNS}	v_{IP}	v_{LB}^*	v_{LB}	BB	CC	t_{CC}	SNC	t_{SNC}
M-n101-k10-C51-V5	3	2492.03	1.50	542	542	529.39	542.00	4183	1729	16.88	582	1.16
M-n121-k7-C61-V4	5	21600.30	2.15	719	-	691.60	707.67	5291	2919	67.26	736	1.97
M-n151-k12-C76-V6	5	21600.60	3.24	659	-	614.49	629.92	2637	4441	106.63	622	1.40
M-n200-k16-C100-V8	12	21601.40	5.34	791	-	734.09	744.86	606	5419	85.24	551	0.83
G-n262-k25-C131-V12	3	21606.00	6.22	3249	-	2863.48	2863.48	0	4251	9.42	268	0.00

Table 5 Computational results for instances generated using $\theta = 3$ and $\gamma = 1/m$

Instance	n'	T	t_{LNS}	v_{LNS}	v_{IP}	v_{LB}^*	v_{LB}	BB	CC	t_{CC}	SNC	t_{SNC}
M-n101-k10-C34-V4	6	5237.69	0.86	458	458	439.46	458.00	17916	413	21.59	1058	3.83
M-n121-k7-C41-V3	11	3790.99	1.19	527	527	507.76	527.00	4041	1150	16.35	843	1.49
M-n151-k12-C51-V4	6	21600.50	1.89	483	-	443.62	465.59	3402	2375	55.20	1146	2.06
M-n200-k16-C67-V6	13	21600.30	3.03	605	-	545.34	563.13	1719	2843	61.71	959	1.63
G-n262-k25-C88-V9	8	21601.80	4.92	2476	-	2065.58	2102.38	181	3560	32.19	764	0.51

The branch-and-cut algorithm was able to solve to optimality three of the instances (one with $\theta = 2$ and two with $\theta = 3$) out of the twelve instances tested in this section, within the time limit imposed on the running time of the algorithms. The optimality gaps as calculated using equation (37) for the instances shown in Tables 4 and 5 that could not be solved to optimality range from 1.58% (for the instance M-n121-k7-C61-V4) to 15.1% (for the instance G-n262-k25-C88-V9). We note that for some instances quite some time is spent by CPLEX adding valid inequalities and solving linear programs. In fact, for the instance named G-n262-k25-C131-V12, the branch-and-cut algorithm terminated in the cut separating phase at the root node due to the time limit, even before branching had begun.

6.6. Optimal Solution of the Ghiani and Improta (2000) Instance

The only existing (published) instance for the GVRP is that proposed by Ghiani and Improta (2000), derived from an instance taken from Araque et al. (1994) with 50 vertices, 25 clusters and 4 vehicles. The solution as reported in Ghiani and Improta (2000) is obtained by transforming the problem into CARP, which is then solved by a heuristic procedure to yield an objective function value of 532.73 (we note for this specific instance that the distances are *not* rounded to the nearest integer). The same instance is solved by Kara and Bektaş (2003) using their proposed formulation to obtain the optimal solution for the first time. The formulation solved by CPLEX 6.0 on a Pentium 1100Mhz PC with 1 GB RAM required 17600.85 CPU seconds. Our recent attempt in solving the same instance on our machine with the formulation of Kara and Bektaş (2003) using CPLEX 10.0 demanded a similar computational time in obtaining an optimal solution, suggesting that such an approach will be unable to cope with larger GVRP instances.

The proposed branch-and-cut algorithm in this paper based on \mathcal{F}_4 solved the Ghiani and Improta (2000) instance, to optimality, in 2.7 CPU seconds, yielding an objective function value of 527.8126996, coinciding with the optimal solution value as reported by Kara and Bektaş (2003). For the solution of this particular instance, the branch-and-cut algorithm terminated with 9 nodes, having separated 43 same-vertex inequalities and 199 capacity constraints.

7. Conclusions

This paper has presented four formulations for the Generalized Vehicle Routing Problem, of which two are polynomial in size and the other two are based on exponential sets of inequalities. The latter two are directed and undirected formulations. Branch-and-cut algorithms have been described for the solution of formulations that are exponential in size. The four formulations have been compared against one another, both analytically and empirically. Extensive computational experiments have

been performed, the results of which show that a branch-and-cut algorithm based on the undirected exponential-sized formulation (\mathcal{F}_4) significantly outperforms the remaining three formulations, as well as the existing approaches, in efficiently solving a wide range of GVRP instances. The proposed algorithm is able to solve instances with up to 101 vertices and 51 clusters, to optimality, within reasonable computational times.

References

- Araque, J.R., G. Kudva, T.L. Morin, J.F. Pekny. 1994. A branch and cut algorithm for vehicle routing problems. *Annals of Operations Research* **50** 37–59.
- Bautista, J., E. Fernández, J. Pereira. 2008. Solving an urban waste collection problem using ants heuristics. *Computers & Operations Research* **35** 3020–3033.
- Cordeau, J.-F., G. Laporte. 2006. Modeling and optimization of vehicle routing and arc routing problems. L. Pitsoulis G. Appa, H.P. Williams, eds., *Transportation, Handbook on Modelling for Discrete Optimization*, vol. 14. Springer, New York, 151–191.
- Cordeau, J.-F., G. Laporte, Savelsbergh M.W.P., D. Vigo. 2007. Vehicle routing. C. Barnhart, G. Laporte, eds., *Transportation, Handbooks in Operations Research and Management Science*, vol. 14. Elsevier, Amsterdam, 367–428.
- Fischetti, M., J.J.S. Gonzalez, P. Toth. 1995. The symmetric generalized traveling salesman polytope. *Networks* **26** 113–123.
- Fischetti, M., Gonzalez J.J.S., P. Toth. 1997. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* **45** 378–394.
- Gavish, B., S.C. Graves. 1978. The traveling salesman problem and related problems. Tech. Rep. OR 078-78, Massachusetts Institute of Technology.
- Ghiani, G., G. Improta. 2000. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research* **122** 11–17.
- Gouveia, L. 1995. A result on projection for the Vehicle Routing Problem. *European Journal of Operational Research* **85** 610–624.
- Kara, I., T. Bektaş. 2003. Integer linear programming formulation of the generalized vehicle routing problem. Presented at the EURO/INFORMS Joint International Meeting, Istanbul, July 06-10, Türkiye.
- Kara, I., G. Laporte, T. Bektaş. 2003. A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research* **158** 793–795.
- Kirkpatrick, S., Jr. C.D. Gelatt, M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* **220**(4598) 671–680.

- Laporte, G. 2007. What you should know about the vehicle routing problem. *Naval Research Logistics* **54** 811–819.
- Letchford, A.N., J.-J. Salazar-Gonzalez. 2006. Projection results for vehicle routing. *Mathematical Programming* **105** 251–274.
- Lysgaard, J. 2003. Cvrpsep: A package of separation routines for the capacitated vehicle routing problem. Tech. Rep. Working paper 03-04, Department of Management Science and Logistics, Aarhus School of Business, Denmark.
- Martello, S., P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley.
- Miller, C.E., A.W. Tucker, R.A. Zemlin. 1960. Integer programming formulations and traveling salesman problems. *Journal of Association for Computing Machinery* **7** 326–329.
- Pisinger, D., S. Ropke. 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research* **34** 2403–2435.
- Ropke, S., D. Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* **40** 455–472.
- Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, *Lecture Notes in Computer Science*, vol. 1520. Springer, Berlin, 417–431.

Appendix A: Detailed Results of the Computational Experiments

The following nomenclature is used in the tables.

- Instance: name of the instance,
- n' : number of vertices removed by the preprocessing procedure,
- T : the total computational time (in CPU seconds) required by the branch-and-cut algorithm to solve the corresponding instance,
- t_{LNS} : the time (in CPU seconds) spent by the LNS heuristic,
- v_{LNS} : value of the best solution as output by the LNS heuristic,
- v_{IP} : value of the best integer solution as output by the branch-and-cut algorithm,
- v_{LB}^r : value of the lower bound of the root node after adding violated inequalities (27) and (28) and after CPLEX has added inequalities of its own,
- v_{LB} : value of the best lower bound in the branch-and-cut tree,
- BB: number of nodes in the branch-and-cut tree,
- CC : number of violated capacity inequalities (28),
- t_{CC} : total time (in CPU seconds) required to separate inequalities (28),
- SNC : number of violated same-vertex inequalities (27),
- t_{SNC} : total time (in CPU seconds) required to separate the violated same-vertex inequalities (27).

Table 6 Computational results for instances generated using $\theta = 2$ and $\gamma = 1/m$

Instance	n'	T	t_{LNS}	v_{LNS}	v_{IP}	v_{LB}^r	v_{LB}	BB	CC	t_{CC}	SNC	t_{SNC}
A-n32-k5-C16-V2	2	113.15	0.26	519	519	474.00	519.00	1847	394	1.25	287	0.06
A-n33-k5-C17-V3	1	1.64	0.29	451	451	437.69	451.00	38	79	0.06	57	0.01
A-n33-k6-C17-V3	4	0.73	0.28	465	465	462.12	465.00	5	69	0.00	24	0.00
A-n34-k5-C17-V3	4	0.84	0.29	489	489	486.36	489.00	5	58	0.00	38	0.00
A-n36-k5-C18-V2	0	31.48	0.32	505	505	480.21	505.00	612	141	0.55	212	0.05
A-n37-k5-C19-V3	3	0.76	0.35	432	432	430.40	432.00	1	47	0.01	25	0.00
A-n37-k6-C19-V3	1	28.20	0.32	584	584	559.04	584.00	264	306	0.39	216	0.01
A-n38-k5-C19-V3	1	2.96	0.34	476	476	463.22	476.00	27	55	0.00	60	0.00
A-n39-k5-C20-V3	3	45.57	0.36	557	557	530.58	557.00	544	298	0.68	278	0.01
A-n39-k6-C20-V3	3	4.88	0.37	544	544	525.80	544.00	42	151	0.13	62	0.00
A-n44-k6-C22-V3	3	23.22	0.37	608	608	572.73	608.00	210	347	0.51	156	0.01
A-n45-k6-C23-V4	6	6.79	0.44	613	613	595.67	613.00	112	203	0.14	71	0.00
A-n45-k7-C23-V4	4	1465.19	0.42	674	674	630.86	674.00	3184	1571	11.04	359	0.21
A-n46-k7-C23-V4	1	10.23	0.43	593	593	573.95	593.00	43	232	0.14	96	0.00
A-n48-k7-C24-V4	5	299.80	0.44	667	667	630.35	667.00	1829	843	3.71	261	0.15
A-n53-k7-C27-V4	4	15.92	0.51	603	603	589.48	603.00	40	339	0.24	97	0.01
A-n54-k7-C27-V4	1	68.29	0.49	690	690	665.31	690.00	372	633	1.27	273	0.08
A-n55-k9-C28-V5	4	82.60	0.53	699	699	668.04	699.00	577	755	1.71	182	0.06
A-n60-k9-C30-V5	4	75.60	0.59	769	769	750.75	769.00	215	857	0.97	143	0.01
A-n61-k9-C31-V5	5	43.71	0.61	638	638	621.03	638.00	243	682	1.13	160	0.02
A-n62-k8-C31-V4	3	122.73	0.61	740	740	722.34	740.00	210	735	1.11	231	0.06
A-n63-k10-C32-V5	3	4355.24	0.60	801	801	759.56	801.00	5430	2544	39.21	462	0.47
A-n63-k9-C32-V5	1	7200.07	0.62	912	-	864.77	900.29	4749	1627	29.21	500	0.56
A-n64-k9-C32-V5	3	1204.27	0.63	763	763	733.94	763.00	1831	1402	9.08	489	0.28
A-n65-k9-C33-V5	6	28.95	0.68	682	682	665.09	682.00	54	734	0.49	94	0.00
A-n69-k9-C35-V5	10	817.90	0.76	680	680	648.02	680.00	2569	1515	14.51	274	0.30
A-n80-k10-C40-V5	3	7200.03	1.00	997	998	916.09	957.36	4487	1834	38.66	574	0.73
B-n31-k5-C16-V3	4	0.13	0.29	441	441	441.00	441.00	0	30	0.00	15	0.00
B-n34-k5-C17-V3	5	0.08	0.30	472	472	472.00	472.00	0	23	0.01	16	0.00
B-n35-k5-C18-V3	3	0.08	0.34	626	626	626.00	626.00	0	18	0.00	20	0.01
B-n38-k6-C19-V3	3	0.69	0.32	451	451	450.82	451.00	3	62	0.00	21	0.00
B-n39-k5-C20-V3	4	0.20	0.39	357	357	356.50	357.00	2	27	0.01	23	0.00
B-n41-k6-C21-V3	7	2.62	0.36	481	481	472.19	481.00	79	137	0.11	74	0.01
B-n43-k6-C22-V3	3	9.22	0.38	483	483	472.11	483.00	82	272	0.13	88	0.00
B-n44-k7-C22-V4	5	3.26	0.39	540	540	537.08	540.00	17	161	0.04	65	0.01
B-n45-k5-C23-V3	2	0.60	0.45	497	497	496.62	497.00	5	33	0.02	52	0.00
B-n45-k6-C23-V4	6	53.71	0.44	478	478	466.72	478.00	717	392	0.87	154	0.05
B-n50-k7-C25-V4	6	0.56	0.53	449	449	446.29	449.00	23	35	0.02	39	0.00
B-n50-k8-C25-V5	1	3249.18	0.49	916	916	890.86	916.00	7180	1936	30.33	598	0.49
B-n51-k7-C26-V4	7	0.44	0.46	651	651	650.51	651.00	4	39	0.00	29	0.00
B-n52-k7-C26-V4	9	0.12	0.50	450	450	450.00	450.00	0	38	0.00	15	0.00
B-n56-k7-C28-V4	8	2.98	0.56	486	486	483.44	486.00	18	204	0.07	61	0.01
B-n57-k7-C29-V4	11	1.82	0.54	751	751	748.43	751.00	21	129	0.03	43	0.00
B-n57-k9-C29-V5	5	21.95	0.54	942	942	933.43	942.00	115	582	0.52	170	0.00
B-n63-k10-C32-V5	5	12.23	0.61	816	816	806.70	816.00	75	347	0.20	119	0.00
B-n64-k9-C32-V5	8	0.78	0.67	509	509	507.80	509.00	3	67	0.00	46	0.00
B-n66-k9-C33-V5	4	14.42	0.69	808	808	802.43	808.00	34	520	0.22	108	0.02
B-n67-k10-C34-V5	4	35.76	0.68	673	673	663.37	673.00	229	627	0.88	170	0.05
B-n68-k9-C34-V5	10	9.21	0.69	704	704	700.92	704.00	27	364	0.10	81	0.00
B-n78-k10-C39-V5	3	248.21	0.80	803	803	791.44	803.00	570	1051	3.16	307	0.09
P-n16-k8-C8-V5	3	0.02	0.16	239	239	239.00	239.00	0	20	0.00	6	0.00
P-n19-k2-C10-V2	3	0.01	0.19	147	147	147.00	147.00	0	4	0.00	4	0.00
P-n20-k2-C10-V2	3	0.01	0.19	154	154	154.00	154.00	0	6	0.00	9	0.00
P-n21-k2-C11-V2	2	0.04	0.20	160	160	160.00	160.00	0	8	0.00	8	0.00
P-n22-k2-C11-V2	2	0.11	0.20	162	162	160.65	162.00	3	16	0.00	22	0.00
P-n22-k8-C11-V5	1	0.04	0.22	314	314	314.00	314.00	0	34	0.01	14	0.00
P-n23-k8-C12-V5	5	0.75	0.20	312	312	303.10	312.00	17	108	0.02	19	0.00
P-n40-k5-C20-V3	3	2.09	0.40	294	294	284.32	294.00	29	89	0.03	82	0.02
P-n45-k5-C23-V3	2	2.18	0.45	337	337	330.84	337.00	16	75	0.04	64	0.00
P-n50-k10-C25-V5	2	1162.92	0.47	410	410	377.97	410.00	2715	1224	7.68	330	0.23
P-n50-k7-C25-V4	2	26.73	0.52	353	353	337.11	353.00	387	327	0.73	161	0.03
P-n50-k8-C25-V4	2	7200.07	0.46	392	-	342.80	378.38	9415	2012	42.29	450	0.61
P-n51-k10-C26-V6	1	38.75	0.49	427	427	405.14	427.00	213	716	0.94	144	0.03
P-n55-k10-C28-V5	2	1536.69	0.52	415	415	387.72	415.00	4623	1387	16.00	372	0.32
P-n55-k15-C28-V8	2	7200.08	0.53	555	-	508.65	545.32	4537	2956	38.79	367	0.26
P-n55-k7-C28-V4	2	125.24	0.57	361	361	342.69	361.00	967	579	2.30	218	0.07
P-n55-k8-C28-V4	2	38.87	0.58	361	361	347.79	361.00	359	258	0.57	171	0.05
P-n60-k10-C30-V5	2	7200.04	0.58	443	-	406.04	433.03	7048	2324	44.77	489	0.74
P-n60-k15-C30-V8	3	7200.17	0.63	565	-	522.22	553.88	5122	3379	52.22	337	0.45
P-n65-k10-C33-V5	0	1805.45	0.67	487	487	461.28	487.00	2951	1565	15.28	420	0.34
P-n70-k10-C35-V5	1	175.75	0.75	485	485	468.80	485.00	405	1126	3.12	229	0.04
P-n76-k4-C38-V2	2	25.83	0.95	383	383	374.86	383.00	108	313	0.45	177	0.08
P-n76-k5-C38-V3	2	16.23	0.92	405	405	396.56	405.00	108	311	0.36	158	0.05
P-n101-k4-C51-V2	1	169.23	1.87	455	455	442.87	455.00	647	612	2.36	403	0.26

Table 7 Computational results for instances generated using $\theta = 3$ and $\gamma = 1/m$

Instance	n'	T	t_{LNS}	v_{LNS}	v_{IP}	v_{LB}^r	v_{LB}	BB	CC	t_{CC}	SNC	t_{SNC}
A-n32-k5-C11-V2	6	0.11	0.20	386	386	380.33	386.00	5	19	0.02	24	0.00
A-n33-k5-C11-V2	7	0.46	0.20	318	315	306.80	315.00	7	22	0.00	31	0.00
A-n33-k6-C11-V2	7	1.23	0.19	370	370	355.12	370.00	23	47	0.02	81	0.00
A-n34-k5-C12-V2	5	1.66	0.20	419	419	408.14	419.00	26	45	0.01	61	0.00
A-n36-k5-C12-V2	7	1.26	0.21	396	396	367.30	396.00	81	22	0.00	83	0.00
A-n37-k5-C13-V2	7	0.67	0.24	347	347	344.43	347.00	3	26	0.00	43	0.01
A-n37-k6-C13-V2	3	19.40	0.23	431	431	390.83	431.00	309	108	0.22	294	0.02
A-n38-k5-C13-V2	6	0.72	0.23	367	367	362.94	367.00	3	29	0.00	36	0.00
A-n39-k5-C13-V2	5	4.55	0.24	364	364	331.71	364.00	150	75	0.12	126	0.02
A-n39-k6-C13-V2	7	1.16	0.23	403	403	388.92	403.00	5	64	0.02	37	0.00
A-n44-k6-C15-V2	3	323.65	0.27	503	503	448.92	503.00	2019	313	1.64	837	0.18
A-n45-k6-C15-V3	9	2.88	0.29	474	474	449.68	474.00	46	71	0.06	100	0.00
A-n45-k7-C15-V3	6	7.44	0.30	475	475	451.43	475.00	69	106	0.12	154	0.00
A-n46-k7-C16-V3	6	22.67	0.30	462	462	424.22	462.00	349	138	0.30	242	0.02
A-n48-k7-C16-V3	8	18.96	0.30	451	451	421.72	451.00	304	133	0.26	237	0.01
A-n53-k7-C18-V3	7	5.93	0.38	440	440	417.52	440.00	85	69	0.09	151	0.00
A-n54-k7-C18-V3	5	57.40	0.37	482	482	441.93	482.00	430	167	0.49	346	0.02
A-n55-k9-C19-V3	7	14.14	0.36	473	473	453.69	473.00	72	158	0.26	143	0.02
A-n60-k9-C20-V3	7	885.22	0.39	595	595	543.49	595.00	2884	632	5.34	771	0.19
A-n61-k9-C21-V4	10	14.45	0.42	473	473	445.40	473.00	160	255	0.23	179	0.00
A-n62-k8-C21-V3	4	859.62	0.42	596	596	556.00	596.00	2532	479	5.39	786	0.26
A-n63-k10-C21-V4	10	279.68	0.41	593	593	550.22	593.00	1541	583	2.87	402	0.10
A-n63-k9-C21-V3	5	7200.05	0.39	642	-	578.91	625.63	8483	945	24.13	1169	0.87
A-n64-k9-C22-V3	5	22.37	0.43	536	536	516.09	536.00	79	166	0.28	280	0.01
A-n65-k9-C22-V3	13	21.88	0.39	500	500	465.19	500.00	174	344	0.40	197	0.01
A-n69-k9-C23-V3	10	4752.37	0.44	520	520	464.76	520.00	10201	1049	25.40	823	0.84
A-n80-k10-C27-V4	4	7200.05	0.56	710	-	629.97	679.43	4813	719	16.34	1101	0.76
B-n31-k5-C11-V2	6	0.21	0.20	356	356	355.92	356.00	2	16	0.00	26	0.00
B-n34-k5-C12-V2	7	0.04	0.21	369	369	369.00	369.00	0	11	0.00	15	0.00
B-n35-k5-C12-V2	6	0.20	0.21	501	501	500.74	501.00	1	15	0.00	33	0.00
B-n38-k6-C13-V2	4	1.30	0.23	370	370	362.76	370.00	33	39	0.01	90	0.01
B-n39-k5-C13-V2	14	0.04	0.23	280	280	280.00	280.00	0	16	0.01	15	0.00
B-n41-k6-C14-V2	11	0.97	0.23	407	407	402.72	407.00	14	28	0.01	52	0.01
B-n43-k6-C15-V2	5	0.58	0.28	343	343	343.00	343.00	0	36	0.00	47	0.00
B-n44-k7-C15-V3	10	1.50	0.28	395	395	388.43	395.00	41	44	0.01	58	0.00
B-n45-k5-C15-V2	7	0.90	0.29	422	410	409.25	410.00	6	22	0.01	41	0.00
B-n45-k6-C15-V2	7	4.76	0.27	336	336	332.35	336.00	24	56	0.04	98	0.00
B-n50-k7-C17-V3	9	0.20	0.31	393	393	393.00	393.00	0	30	0.00	32	0.00
B-n50-k8-C17-V3	7	29.35	0.29	598	598	581.34	598.00	250	169	0.22	239	0.02
B-n51-k7-C17-V3	11	0.39	0.31	511	511	510.87	511.00	4	24	0.00	46	0.00
B-n52-k7-C18-V3	19	0.04	0.34	359	359	359.00	359.00	0	28	0.00	12	0.00
B-n56-k7-C19-V3	12	23.46	0.37	356	356	342.98	356.00	656	55	0.25	238	0.02
B-n57-k7-C19-V3	14	0.87	0.36	558	558	558.00	558.00	0	44	0.01	63	0.00
B-n57-k9-C19-V3	8	471.61	0.35	681	681	664.30	681.00	2699	316	3.23	687	0.22
B-n63-k10-C21-V3	8	11.28	0.39	599	599	591.23	599.00	65	104	0.08	113	0.00
B-n64-k9-C22-V4	11	2.38	0.45	452	452	448.37	452.00	26	52	0.01	89	0.00
B-n66-k9-C22-V3	17	103.46	0.40	609	609	585.52	609.00	1063	452	1.32	290	0.08
B-n67-k10-C23-V4	5	7.21	0.48	558	558	551.24	558.00	72	110	0.05	161	0.00
B-n68-k9-C23-V3	17	109.95	0.44	523	523	507.79	523.00	1250	286	1.05	305	0.05
B-n78-k10-C26-V4	9	8.46	0.53	606	606	601.06	606.00	11	198	0.13	115	0.00
P-n16-k8-C6-V4	7	0.00	0.13	170	170	170.00	170.00	0	2	0.00	1	0.00
P-n19-k2-C7-V1	3	0.02	0.12	111	111	111.00	111.00	0	6	0.00	12	0.00
P-n20-k2-C7-V1	5	0.24	0.12	117	117	113.81	117.00	7	8	0.00	28	0.00
P-n21-k2-C7-V1	4	0.17	0.12	117	117	115.69	117.00	1	10	0.00	15	0.00
P-n22-k2-C8-V1	5	0.05	0.15	111	111	111.00	111.00	0	7	0.00	14	0.00
P-n22-k8-C8-V4	4	0.07	0.17	249	249	249.00	249.00	0	12	0.01	13	0.00
P-n23-k8-C8-V3	6	0.05	0.16	174	174	174.00	174.00	0	20	0.01	12	0.00
P-n40-k5-C14-V2	6	1.13	0.25	213	213	208.35	213.00	12	35	0.02	58	0.00
P-n45-k5-C15-V2	6	11.14	0.29	238	238	210.76	238.00	230	79	0.14	152	0.01
P-n50-k10-C17-V4	5	5.02	0.29	292	292	277.41	292.00	40	72	0.05	108	0.02
P-n50-k7-C17-V3	5	6.43	0.32	261	261	246.92	261.00	110	68	0.10	158	0.01
P-n50-k8-C17-V3	5	7.40	0.29	262	262	248.55	262.00	53	137	0.09	132	0.01
P-n51-k10-C17-V4	4	117.61	0.31	309	309	272.36	309.00	1483	244	1.16	321	0.16
P-n55-k10-C19-V4	4	18.07	0.37	301	301	280.48	301.00	217	152	0.26	182	0.03
P-n55-k15-C19-V6	4	36.03	0.35	378	378	350.60	378.00	195	434	0.60	176	0.06
P-n55-k7-C19-V3	4	78.24	0.37	271	271	243.81	271.00	819	147	0.74	315	0.07
P-n55-k8-C19-V3	4	53.58	0.38	274	274	247.11	274.00	580	119	0.40	279	0.04
P-n60-k10-C20-V4	7	282.66	0.42	325	325	298.82	325.00	2227	379	2.52	386	0.17
P-n60-k15-C20-V5	6	7200.02	0.37	382	-	337.95	379.25	11507	1529	34.75	882	0.88
P-n65-k10-C22-V4	2	1028.22	0.43	372	372	338.33	372.00	4237	596	6.26	527	0.39
P-n70-k10-C24-V4	3	1468.25	0.50	385	385	354.46	385.00	5541	534	8.56	725	0.73
P-n76-k4-C26-V2	3	122.51	0.62	320	309	287.90	309.00	840	152	0.73	478	0.17
P-n76-k5-C26-V2	3	90.11	0.61	309	309	287.03	309.00	561	252	0.75	415	0.08
P-n101-k4-C34-V2	1	6581.79	0.96	374	370	344.87	370.00	13879	328	22.16	1361	3.15