Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation

An Exact Algorithm and a Metaheuristic for the Generalized Vehicle Routing Problem

Minh Hoang Ha
Nathalie Bostel
André Langevin
Louis-Martin Rousseau

November 2012

CIRRELT-2012-73

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

An Exact Algorithm and a Metaheuristic for the Generalized Vehicle Routing Problem

Min Hoang Ha[1], Nathalie Bostel[2], André Langevin[1], Louis-Martin Rousseau[1,*]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

[2] Université de Nantes, I.U.T.-St-Nazaire, 58, rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France

Abstract. The *generalized vehicle routing problem* (GVRP) involves finding a minimum length set of vehicle routes passing through a set of clusters, where each cluster contains a number of vertices, such that the tour includes exactly one vertex from each cluster and satisfies capacity constraints. We consider a version of the GVRP where the number of vehicles is a decision variable. This paper introduces a new mathematical formulation based on a two-commodity flow model. We solve the problem using a branch-and-cut algorithm and a metaheuristic that is a hybrid of the *greedy randomized adaptive search procedure* (GRASP) and the *evolutionary local search* (ELS) proposed in [18]. We perform computational experiments on instances from the literature to demonstrate the performance of our algorithms.

Keywords. Generalized vehicle routing, two-commodity flow model, branch-and-cut, metaheuristic.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

* Corresponding author: Louis-Martin.Rousseau@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1. Introduction

The *capacitated vehicle routing problem* (CVRP) is one of the most popular and challenging combinatorial optimization problems. It involves finding the optimal set of routes for a fleet of vehicles that serves a given set of customers. In classical transportation problems, each customer is served from only one vertex. Therefore, there is always a well-defined set of vertices that must be visited, and we need to find the solution from this set. However, in many real applications a customer can be served from more than one vertex, and the resulting problems are more complex. The GVRP is a generalization of the CVRP and also an extension of the *generalized traveling salesman problem* (GTSP). The GVRP can model problems concerned with the design of bilevel transportation networks; see [6] and [16] for further information on its applications.

The GVRP is defined as follows. Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edge set. $V = \{v_0, ..., v_{n-1}\}$ is the set of n vertices that can be visited, and vertex v_0 is the depot, containing m identical vehicles with a common capacity Q. $C = \{C_0, ..., C_{K-1}\}$ is the set of K clusters. Each cluster C_i except C_0, which contains only the depot, has a demand D_i. Each cluster includes a number of vertices of V, and every vertex in V belongs to exactly one cluster. For each $v_i \in V$, let $\alpha(i)$ be the cluster that contains vertex v_i. The term $D(S) = \sum_{i|C_i \subseteq S} D_i$ is used to represent the total demand in set S which is a subset of V. The number of vehicles m can be constant or variable. A length c_{ij} is associated with each edge of $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$. The GVRP consists in finding m vehicle routes such that (i) each route begins and ends at the depot; (ii) each route visits exactly one vertex of each cluster and visits it only once; (iii) the demand served by each route does not exceed the vehicle capacity Q; and (iv) the total cost is minimized.

The GVRP is clearly NP-hard since it reduces to a VRP when each cluster includes only one vertex or to a GTSP when the capacity constraints are relaxed. The number of papers on this topic is quite limited. The problem was first introduced by Ghiani and Improta [8]. In 2003, Kara and Bektaş [9] proposed the first formulation that was polynomial in the number of constraints and variables. An ant colony algorithm and a genetic algorithm were proposed in [17] and [14] respectively. Recently, Bektaş et al. [6] proposed four formulations and four branch-and-cut algorithms. They concluded that the best formulation was an undirected two-index flow one based on an exponential number of constraints. They also proposed a heuristic based on *large neighborhood search* (LNS) to provide upper bounds for the branch-and-cut algorithms. At about the same time, Pop et al. [16] introduced two new formulations. The first, the node formulation, is similar to the formulation in [9] but produces a stronger lower bound, and the second is a flow-based formulation. The authors directly solved one instance from [8] using CPLEX. They reported no further computational experience with the proposed formulations and did not develop branch-and-cut algorithms.

In this paper, we consider a version of the GVRP where the number of vehicles is a decision variable. This version has not been investigated in the literature. We make two contributions: i) we present a new formulation for the GVRP, and ii) we propose an exact method and a metaheuristic to solve the problem. Computational experiments show that our exact approach can solve instances of up to 121 vertices and 51 clusters, and our metaheuristic gives high-quality solutions for the instances tested in a reasonable computational time. Compared to the formulation proposed in [6], our formulation provides better lower bounds and better performance of the branch-and-cut algorithm.

The remainder of the paper is organized as follows. Section 2 describes our formulation and several valid inequalities. The branch-and-cut algorithm and the metaheuristic are presented in Sections 3 and 4 respectively. Section 5 discusses the computational results, and Section 6 summarizes our conclusions.

2. New formulation for the GVRP

We first reintroduce the best formulation of [6]. Note that Bektaş et al. [6] tackle the case in which the number of vehicles is constant. To adapt it to the context where the number of vehicles is variable, we simply consider the number of vehicles m in the formulation as a decision variable. This formulation uses integer variables $z_{ij}, (v_i, v_j) \in E$ that count the number of times the edge $\{v_i, v_j\}$ is used. Let $\delta(S) = \{(v_i, v_j) \in E : v_i \in S, v_j \notin S\}$ and $z(F) = \sum_{(v_i,v_j)\in F} z_{ij}$ where F is a subset of E. Then the formulation is as follows:

$$\text{Minimize} \quad \sum_{\{v_i,v_j\}\in E} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad z\Big(\delta(C_k)\Big) = 2 \quad \forall C_k \in C \setminus C_0 \tag{2}$$

$$z\Big(\delta(C_0)\Big) = 2m \tag{3}$$

$$z\Big(\delta(S)\Big) + 2 \sum_{\{v_i,v_j\}\in L: i\notin S} z(\{i\} : C_j) \leq 2 \quad \forall C_k \in C \setminus C_0, S \subseteq C_k, L \in \overline{L}_k \tag{4}$$

$$\sum_{(v_i\in S_1, v_j\in S_2)} z_{ij} \leq |S| - \left\lceil \frac{D(S)}{Q} \right\rceil \quad \forall S_1 \subseteq S, \; S_2 \subseteq S, S \subseteq C, |S| \geq 2 \tag{5}$$

$$z_{ij} = 0,1,2 \quad \forall\{v_i,v_j\} \in \delta(0) \tag{6}$$

$$z_{ij} = 0,1 \quad \forall\{v_i,v_j\} \in E \setminus \delta(0) \tag{7}$$

$$m \in \mathbb{N}. \tag{8}$$

In this formulation, constraints (2) ensure that each cluster is visited exactly once. Constraints (3) imply that m vehicles will leave the depot. Constraints (4) are referred to as same-vertex inequalities; they ensure that when a vehicle arrives at a certain vertex in a cluster, it will depart from the same vertex. Here, $\overline{L}_k = \{L : L \subseteq \cup_{i\in C_k} L_i, |L \cap L_i| = 1, \forall i \in C_k\}$ where $L_i = \{i\} \times (C \setminus \{0, \alpha(i)\})$, defined for all $v_i \in V \setminus v_0$. Constraints (5) are the capacity constraints.

We now describe a new integer programming formulation for the GVRP. The idea underlying this formulation was first introduced by [7] for the traveling salesman problem (TSP). Langevin et al. [11] extended this approach to the TSP with time windows. Baldacci et al. [3] used it to derive a new formulation and a branch-and-cut algorithm for the VRP, and Baldacci et al. [2] adapted it for the *covering tour problem* (CTP) without capacity constraints. Currently, together with the two-index flow formulation and the set partitioning formulation, this is one of the most successful formulations underlying exact methods for the CVRP (see [4]).

Our formulation is an extension of that proposed by Baldacci et al. [3] for the CVRP. To adapt this idea for the GVRP, we assume that each vertex v_i of V has a demand d_i equal to the demand $D_{\alpha(i)}$ of the cluster to which it belongs. In other words, all the vertices in a cluster have the same demand as that of the cluster. The difference from CVRP is that we do not need to visit all the vertices of V.

We first extend the original graph G to $\overline{G} = (\overline{V}, \bar{E})$ by adding a new vertex v_n, which is a copy of the depot v_0. We now have $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E \cup \{(v_i, v_n), v_i \in V'\}$, and $c_{in} = c_{0i}\ \forall\, v_i \in V'$.

This formulation requires two flow variables, f_{ij} and f_{ji}, to represent an edge of a feasible GVRP solution along which the vehicle carries a load of Q units. When the vehicle travels from v_i to v_j, flow f_{ij} represents the load collected and flow f_{ji} represents the empty space of the vehicle (i.e., $f_{ji} = Q - f_{ij}$).

Let x_{ij} be a 0-1 variable equal to 1 if edge $\{v_i, v_j\}$ is used in the solution and 0 otherwise. Let y_i be a binary variable that indicates the use of vertex v_i in the solution. Then the GVRP can be stated as:

$$\text{Minimize} \qquad \sum_{\{v_i,v_j\}\in\overline{E}} c_{ij}x_{ij} \tag{9}$$

$$\text{subject to} \qquad \sum_{v_i\in C_k} y_i = 1 \qquad \forall C_k \in C \tag{10}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{11}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2d_i y_i \qquad \forall v_k \in V' \tag{12}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} d_i y_i \tag{13}$$

$$\sum_{j\in V'} f_{nj} = mQ \tag{14}$$

$$f_{ij} + f_{ji} = Qx_{ij} \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{15}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{16}$$

$$x_{ij} = 0, 1 \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{17}$$

$$y_i = 0, 1 \qquad \forall v_i \in V' \tag{18}$$

$$m \in \mathbb{N}. \tag{19}$$

The objective (9) is to minimize the total travel cost. Constraints (10) ensure that the tour includes exactly one vertex from each cluster, while constraints (11) ensure that each vertex of V' is visited at most once. Constraints (12) to (15) define the flow variables. Specifically, constraints (12) state that the inflow minus the outflow at each vertex $v_i \in V'$ is equal to $2d_i$ if v_i is used and 0 otherwise. The outflow at the source vertex v_0 (13) is equal to the total demand of the vertices that are used in the solution, and the outflow at the sink v_n (14) corresponds to the total capacity of the vehicle fleet. Constraint (15) is derived from the definition of the flow variables. Constraints (16) to (19) define the variables.

Figure 1 shows a feasible solution for the GVRP with five clusters and two routes in the case where $Q = 50$ under the two-commodity form. The demands of clusters $C_1, ..., C_5$ are 10, 20, 20, 20, and 20 respectively. The solid lines in the figure represent the flows f_{ij} and the dotted lines represent the flows f_{ji}.

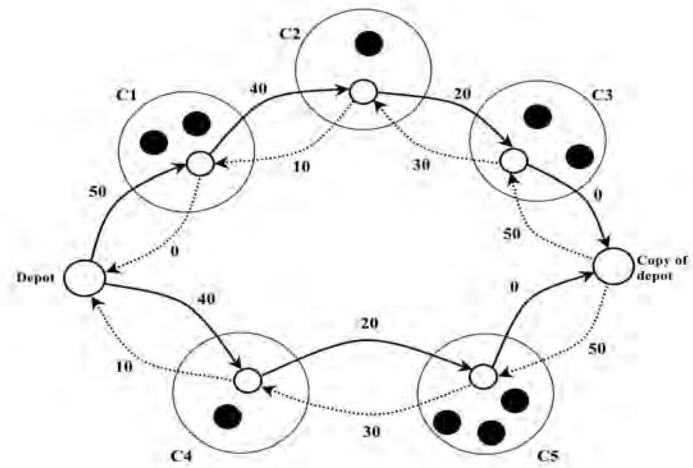

Figure 1: Flow paths for solution with two routes

The linear relaxation of the GVRP can be strengthened by the addition of valid inequalities. The following inequalities are deduced from the definition of the binary variables:

$$x_{ij} \leqslant y_i \text{ and } x_{ij} \leqslant y_j \ (v_i \text{ or } v_j \in V). \tag{20}$$

The following flow inequalities were introduced in [2]:

$$f_{ij} \geq d_j x_{ij}, f_{ji} \geq d_i x_{ij} \text{ if } i,j \neq v_0 \text{ and } i,j \neq v_n. \tag{21}$$

As shown in [6], every GVRP instance can be transformed to a CVRP instance by shrinking each cluster to a single vertex. Therefore, the inequalities that are valid for the CVRP such as the comb, extended comb, capacity, generalized capacity, and hypotour inequalities are also valid for the GVRP. Here, we restrict ourselves to the capacity constraints, originally proposed by [12]:

$$\sum_{(v_i \in S_1, v_j \in S_2)} x_{ij} \leq |S| - \left\lceil \frac{D(S)}{Q} \right\rceil (S_1 \subseteq S, S_2 \subseteq S, S \subseteq C, |S| \geq 2). \tag{22}$$

3. Branch-and-cut algorithm

The GVRP is solved to optimality using a standard branch-and-cut algorithm. We solve a linear program containing the constraints (9), (10), (11), (12), (13), (14), (15), (16), $0 \leq x_{ij}, y_i \leq 1$, and $m \geq 0$. We then search for violated constraints of type (20), (21), and (22), and we add the constraints detected to the current LP, which is then reoptimized. This process is repeated until all the constraints are satisfied. If there are fractional variables, we branch to create two subproblems. If all the variables are integer, we explore another subproblem.

The separation of the constraints of type (20) and (21) is straightforward. To generate the capacity constraints (22), we use the *greedy randomized algorithm*, proposed by [1] and reused in [3].

Our branch-and-cut algorithm is built around CPLEX 12.4 with the Callable Library. As in [6], we enable CPLEX's implementation of strong branching. All the other CPLEX parameters are set to their default values.

We have tested several branching techniques, such as branching on the variables y before x and branching on the variables x before y, but these do not outperform the CPLEX branching. Hence, we let CPLEX make the branching decisions.

4. Metaheuristic

In this section, we present a metaheuristic for the GVRP. It aims to provide good upper bounds for the branch-and-cut algorithm. The main component is a split procedure that converts a giant tour based on the clusters (i.e., each node of this tour is a cluster) and encoded as a TSP tour into a GVRP solution. This procedure is embedded in an algorithm that is a hybrid of GRASP and the ELS method proposed in [18]. GRASP can be considered as a multi-start local search in which each initial solution is built using a greedy randomized heuristic and then improved by local search. In the ELS method, a single solution is mutated to obtain several children that are then improved by local search. The next generation is the best solution among the parent and its children. We now discuss these procedures in detail.

4.1. Split, concat, and mutate procedures

The split procedure is the backbone of our metaheuristic. It splits a giant tour into GVRP routes. This procedure was originally introduced in [5], and it has been integrated into metaheuristics for various vehicle routing problems (see e.g., [10, 15, 19]). The methods based on the split principle are simple and fast, and they give high-quality solutions. Algorithm 1 gives an efficient implementation of split for the GVRP. This implementation is based on that used in [20] for the split for the CVRP.

The input to Algorithm 1 is a permutation S of the K cluster indices. The split procedure constructs an auxiliary graph H with K nodes. Each subsequence of clusters (S_i,...,S_j) that can be considered as one route (the total demand is not greater than Q) is modeled by an arc ($i-1$,j) of graph H. To compute the cost of this arc, we use for each vertex $v_k \in S_t$ $(t = i, ..., j)$ a label $costsum(v_k)$ representing the shortest path from vertex 0 through clusters $S_i, ..., S_{t-1}$ to v_k. The cost of arc ($i - 1$,j) is the smallest value of $costsum(v_k)$ for $v_k \in S_j$ and can be computed in $\mathcal{O}(\lambda^2)$ where λ is the maximum number of vertices of a cluster.

As in [20], Algorithm 1 implements the split procedure without creating the graph H explicitly. We use Bellman's algorithm to calculate the shortest path from vertex 0 to S_k. Each cluster S_k is associated with a label V_k representing the cost of the shortest path from vertex 0 to cluster S_k in H. Instead of storing each arc (i, j) of H, we update label W of node j when necessary. Let b be the maximum number of clusters per route. Prins et al. [20] showed that the complexity of Bellman's algorithm is $\mathcal{O}(bK)$. The cost of each arc is computed in $\mathcal{O}(\lambda^2)$, so our split procedure runs in $\mathcal{O}(bK\lambda^2)$.

The concat procedure simply concatenates GVRP routes into a giant tour. It takes the indices of the clusters that include the vertices of the GVRP

Algorithm 1 Split implementation for GVRP

```
W(0) ⇐ 0;
P(0) ⇐ 0;
for i = 1 → K do
    W(i) ⇐ +∞;
end for
for i = 1 → K do
    j ⇐ i;
    load ⇐ 0;
    repeat
        load ⇐ load + D_{S_j};
        for all v_t ∈ S_j do
            if i = j then
                costsum(v_t) ⇐ 2c_{0v_t};
            else
                costsum(v_t) ⇐ min_{v_e ∈ S_{j−1}} {costsum(v_e) − c_{0v_e} + c_{v_e v_t} + c_{0v_t}};
            end if
        end for
        if load ≤ Q then
            W(j) ⇐ min{W(j); W(i − 1) + min_{v_t ∈ S_j} costsum(v_t)};
        end if
        j ⇐ j + 1;
    until j > K or load > Q
end for
```

solution and removes the copies of the depot node. The mutate procedure randomly swaps the position of two vertices in the giant tour.

4.2. Local search procedure

Our local search consists of classical moves: the one-point move, two-point move, 2-opt move, or-opt move, three-point move, and two-adjacent-point move. The one-point move relocates a vertex, and the two-point move swaps the position of two vertices. The three-point move swaps the position of an edge with the position of another vertex. In the 2-opt move, we remove two edges and replace them with two new edges. In the or-opt move, we relocate a string of two vertices. The two-adjacent-point move swaps the position of two edges. All these moves operate both within routes and between routes.

Tests show that the impact of the order of these searches on the performance of our algorithm is not clear. In our implementation, we use the following order to obtain slightly better results: one-point, or-opt, three-point, two-point, 2-opt, and two-adjacent-point.

It is important to note that in the relocation process a vertex can be replaced by vertices in the same cluster. Therefore, the vertices before and after the exchange may be different (see Fig. 2 for an example of a three-point move).

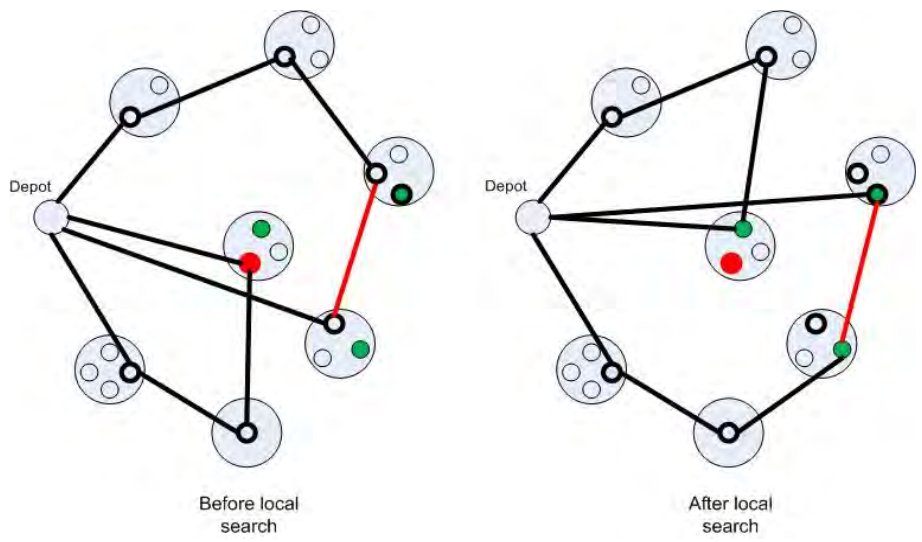

Figure 2: Three-point move for GVRP

4.3. Heuristic for initial solutions

To build an algorithm based on GRASP, we use a randomized version of the nearest-neighbor TSP heuristic to generate a number of GVRP solutions. At each iteration, for a route ending at vertex v_i, we find the nearest uc unrouted clusters. The next vertex of the solution is randomly chosen from the vertices of these uc clusters. Too small a value of uc can limit the algorithm, and too large a value can make the algorithm too random. Our computational tests show that $uc = 2$ gives the best results.

4.4. Metaheuristic algorithm

Algorithm 2 gives the pseudocode for the resulting metaheuristic. S and $f(S)$ denote a solution and its cost respectively. S_{final} and $f(S_{final})$ represent the final results yielded by the metaheuristic. Parameter n_p is the number of phases (each phase generates one final solution for GRASP), n_i the number of iterations per phase, and n_c the number of children generated at each iteration of ELS.

An initial solution S^* is created by the randomized nearest-neighbor heuristic (the RandomInsert procedure) and then improved by the LocalSearch procedure. ELS begins with a giant tour L generated by the concat procedure and performs n_i iterations. At each iteration, we create n_c children of

S by randomly swapping two nodes of the giant tour L (mutate procedure), splitting the giant tour to get a GVRP solution (split procedure), and improving the solution via the LocalSearch procedure. The best child is stored in $\overline{S}$. S^* is updated if the best child is a better solution. The solution S^* is then used for the next ELS iteration.

Algorithm 2 Pseudocode for metaheuristic

```
f(S_final) ⇐ +∞;
for i = 1 → n_p do
    S* ⇐ RandomInsert;
    S* ⇐ LocalSearch(S*);
    L ⇐ Concat(S*);
    for i = 1 → n_i do
        f̄ ⇐ +∞;
        for j = 1 → n_c do
            L ⇐ Mutate(L);
            S ⇐ Split(L);
            S ⇐ LS(S);
            if f(S) < f̄ then
                f̄ ⇐ f(S);
                S̄ ⇐ S;
            end if
        end for
        if f̄ < f(S*) then
            S* ⇐ S̄;
        end if
    end for
    if f(S*) < f(S_final) then
        S_final ⇐ S*;
        f(S_final) ⇐ f(S*);
    end if
end for
```

5. Computational experiments

In this section, we reintroduce the GVRP instances and describe the computational evaluation of the proposed algorithms. Our algorithms are coded in C/C++ and the tests are run on a 2.4-GHz Intel Xeon. We use the effective preprocessing algorithm for the GVRP proposed in [6] to reduce the size of the instances.

The parameters n_p, n_i, and n_c in the metaheuristic are chosen empirically. We have tested many combinations, and the following combination gives the best performance in terms of both quality and computational time for our algorithm: $\{n_p, n_i, n_c\}$=$\{30, 80, 20\}$.

We test our algorithms on the Bektaş instances. They were proposed in [6] and include 158 instances derived from the existing instances A, B, P, M, and G in the CVRP-library. The A, B, and P instances contain 16 to 101 vertices, while the M and G instances are larger with up to 262 vertices. The number of clusters is calculated via $K = \lceil n/\theta \rceil$ with θ=2 and 3.

5.1. Results

This subsection presents the results of our branch-and-cut algorithm and the metaheuristic for the GVRP. The computational time of our exact algorithm is limited, as in [6], to 2 hours for small instances (the Bektaş A, B, and P type instances) and to 6 hours for large instances (the Bektaş M and G type instances). The results are summarized in Table 1, and detailed results can be found in the Appendix. The summary table shows, for each instance, its name (Data), the value of θ (θ), the average computational time for the metaheuristic ($\bar{t}$), and the average computational time for the branch-and-cut algorithm ($\overline{T}$). The $\overline{g}$ column presents the average percentage optimality gap. Let UB be the value of the final solution found by the branch-and-cut algorithm or that found by the metaheuristic (if the branch-and-cut algorithm cannot find a solution). Then the percentage optimality gap g is

$$g = \frac{100(UB - LB)}{UB}. \tag{23}$$

In the next columns, $\overline{BB}$ is the average number of nodes in the branch and bound tree, and $\overline{DO}$, $\overline{FLOW}$, and $\overline{CAP}$ are the average numbers of violated dominance inequalities (20), flow inequalities (21), and capacity inequalities (22) respectively. Finally, `Succ` is the number of instances that were solved to optimality.

The results show that the exact method based on our formulation can solve all instances of type A, B, and P with $\theta = 3$ and of type B with $\theta = 2$. There are two instances of type A and P with $\theta = 2$ for which we cannot find optimal solutions. For the large instances of type M and G, our algorithm can solve two instances with $\theta = 3$ and one with $\theta = 2$. It seems that problems with $\theta = 3$ are easier than those with $\theta = 2$. The problem difficulty increases with n and K. All the types of user cuts are used in the branching process, with the constraints (22) being the most frequently generated (see Tables 6, 7, and 8 in the Appendix).

Our results confirm the high quality of the metaheuristic (see Tables 3, 4, and 5 in the Appendix). The branch-and-cut algorithm provides the optimal solution for 149 instances, and our metaheuristic finds all these solutions. For the instances where the optimal solution is unknown, the branch-and-cut algorithm cannot improve on the metaheuristic. Moreover, its computational time is acceptable, reaching 861.73 s (about 15 min) for the largest instance, G-n262-k25-C88.

Data	θ	$\overline{t}$	$\overline{T}$	$\overline{g}$	$\overline{BB}$	$\overline{DO}$	$\overline{FLOW}$	$\overline{CAP}$	Succ
A	2	26.54	637.67	0.10	1295.1	184.15	507.26	2367.00	26/27
B	2	28.17	33.71	0.00	120.9	109.39	283.17	1170.70	23/23
P	2	35.18	541.58	0.15	1198.7	155.00	391.63	2350.83	23/24
M	2	279.60	16330.32	2.55	2374.3	669.25	1761.50	25552.25	1/4
G	2	822.84	21610.65	10.84	233.0	911.00	2744.00	38569.00	0/1
A	3	23.55	356.38	0.00	1010.9	200.74	429.93	857.30	27/27
B	3	24.16	25.08	0.00	155.9	111.91	245.91	430.52	23/23
P	3	41.59	102.42	0.00	414.0	169.17	306.46	610.58	24/24
M	3	356.71	11918.15	1.78	4943.5	854.75	1767.00	8975.50	2/4
G	3	861.73	21601.39	9.59	508.0	1243.00	3097.00	22258	0/1

Table 1: Summary of computational results

5.2. Comparison of our formulation with the best one proposed in [6]

We now compare our formulation with the best formulation of [6] by comparing the performance of the branch-and-cut algorithms. To generate the constraints (5), Bektaş et al. [6] used the heuristic routines of [13]. For a fair comparison, we used the same procedure to separate the capacity constraints, i.e., the *greedy randomized algorithm* proposed in [1]. We also used the same upper bounds in the branch-and-cut algorithm based on the formulation of [6]. Moreover, we set the parameters of CPLEX 12.4 to the same values. The criteria used for the comparison are the number of successful instances (`Succ`), the computational time (Time), the lower bound at the root node ($LB0$), the best lower bound at the end of the solution process (LB), and the number of nodes in the branch-and-cut tree (BB).

Table 2 summarizes the results of this experiment, and detailed results can be found in the Appendix (Tables 3, 4, and 5). For each criterion we indicate the better result in bold. Sets 1 and 2 are the instances generated with $\theta = 2$ and 3 respectively. As can be seen, our formulation is better for most of the criteria. We solve one more instance of Set 1 (P-n60-k15-C30-V8) and two more instances of Set 2 (A-n63-k9-C21-V3 and A-n80-k10-C27-V4). It is slightly slower on two groups of instances (A with $\theta = 2$ and B with $\theta = 3$) and significantly faster on the other groups. It is also faster in terms of the average computational time for both Set 1 and Set 2 and gives better values for $LB0$ and LB. Our formulation gives better lower bounds at the root node ($LB0$) on 147 of the 158 instances; whereas the Bektaş formulation is better on only 3 instances. Moreover, our branch-and-cut trees have fewer nodes.

Data	θ	Ha et al.					Bektaş et al.				
		Succ	Time	$LB0$	LB	BB	Succ	Time	$LB0$	LB	BB
A	2	26/27	637.67	**615.65**	**633.68**	**1295.1**	26/27	**620.91**	601.10	632.82	2111.0
B	2	23/23	**33.71**	**594.54**	599.30	**120.9**	23/23	48.73	590.74	599.30	530.4
P	2	**23/24**	**541.58**	**348.27**	**357.76**	**1198.7**	22/24	779.83	339.44	357.71	3433.3
M	2	1/4	**16330.31**	**639.62**	**658.66**	**2374.3**	1/4	16813.97	637.32	651.15	5280.8
G	2	0/1	21603.23	**2930.49**	**2945.02**	**233.0**	0/1	21603.67	2809.1	2821.84	328.0
Set 1	2	**73/79**	**1492.59**	**558.79**	**570.37**	**965.2**	72/79	1588.11	548.38	568.12	2190.5
A	3	**27/27**	**356.38**	**453.67**	**471.22**	**1010.9**	25/27	627.66	431.52	468.58	2688.8
B	3	23/23	24.99	**455.42**	459.57	**155.9**	23/23	**15.47**	445.71	459.57	365.9
P	3	24/24	**136.51**	**288.96**	300.94	**414.0**	24/24	191.29	272.75	300.94	1870.5
M	3	2/4	**11918.15**	**495.99**	**508.03**	**4943.5**	2/4	12693.51	479.99	499.65	12010.0
G	3	0/1	21601.39	**2227.79**	**2239.50**	**508.0**	0/1	21601.36	2035.83	2053.25	785.0
Set 2	3	**76/79**	**1037.08**	**417.82**	**428.56**	**742.1**	74/79	1178.75	400.20	424.88	2070.1

Table 2: Comparison with method of [6]

6. Conclusion

We have presented a new integer programming formulation, a branch-and-cut algorithm, and a metaheuristic for the GVRP. We have reported computational results for sets of instances from the literature with up to 262 vertices where the tour contains up to 131 vertices. Our results clearly demonstrate the effectiveness of our approach. The branch-and-cut algorithm based on our formulation is more effective than the current state-of-art algorithm of [6] in terms of the number of successful instances, the quality of the lower bounds, and the number of nodes in the branch-and-bound tree. Our metaheuristic gives high-quality solutions in a reasonable computational time.

Future research directions include applying the exact and heuristic approaches proposed in this paper to other problems such as the arc-counterpart of the GVRP, the *generalized arc routing problem*.

References

[1] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. Rapport de recherche, ARTEMIS-IMAG, Grenoble, France, 1995.

[2] R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for the covering tour problem. In Ramesh Sharda, Stefan Vob, César Rego, and Bahram Alidaee, editors, *Metaheuristic Optimization Via Memory and Evolution*, pages 55–91. Springer Verlag, 2005.

[3] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52:723–738, 2004.

[4] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218:1–6, 2011.

[5] J. E. Beasley. Route first-cluster second methods for vehicle routing. *Omega*, 11:403–408, 1983.

[6] T. Bektaş, G. Erdoğan, and S. Ropke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, 2011.

[7] G. A. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium*, 41:167–178, 1984.

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

[9] I. Kara and T. Bektaş. Integer linear programming formulation of the generalized vehicle routing problem. In *Proceedings of the 5th EURO/INFORMS Joint International Meeting*, 2003.

[10] L. Labadi, C. Prins, and M. Reghioui. A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations Research*, 42:415–431, 2008.

[11] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, and F. Soumis. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, 23:631–640, 1993.

[12] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1058–1073, 1985.

[13] J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working paper 03-04, Department of Management Science and Logistics, Aarhus School of Business, Denmark, 2003.

[14] O. Matei, P. C. Pop, and C. Chira. A genetic algorithm for solving the generalized vehicle routing problem. In E.-S. Corchado Rodriguez et al., editor, *Proceedings of HAIS 2010*, volume 6077 of *Lecture Notes in Artificial Intelligence*, pages 119–126, 2010.

[15] S. U. Ngueveu, C. Prins, and R. Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers and Operations Research*, 37:1877–1885, 2010.

[16] P. C. Pop, I. Kara, and A. H. Marc. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, 36:97–107, 2011.

[17] P. C. Pop, C. M. Pintea, I. Zelina, and D. Dumitrescu. Solving the generalized vehicle routing problem with an ACS-based algorithm. In *Proceedings of BICS 2008*, volume 1117, pages 157–162. American Institute of Physics (AIP), 2009.

[18] C. Prins. A GRASP x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, Studies in Computational Intelligence, pages 35–53. Springer Verlag, 2009.

[19] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22:916–928, 2009.

[20] C. Prins, N. Labadi, and M. Reghiouit. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47:507–535, 2008.

Appendix: Detailed Results of Computational Experiments

In the tables, blank entries indicate that the algorithm did not find a solution. The column headings are as follows:

Data: name of instance;
Ha et al.: branch-and-cut algorithm based on our formulation;
Bektaş et al.: branch-and-cut algorithm based on the formulation of [6];
Node: number of nodes in search tree of branch-and-cut algorithm;
Time: computational time in seconds;
m: number of vehicles in solution;
$LB0$: value of lower bound at root node;
LB: value of best lower bound after branching;
DO: number of constraints of type (20);
$FLOW$: number of constraints of type (21);
CAP: number of constraints of type (22);
SAM: number of constraints of type (4).

	Metaheuristic			Ha et al.				Bektaş et al.			
Data	m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
A-n32-k5-C16	3	11.38	508	15.29	481.88	508.00	205	6.72	464.81	508.00	430
A-n33-k5-C17	3	12.43	451	4.07	447.25	451.00	8	0.83	425.25	451.00	67
A-n33-k6-C17	3	10.08	465	2.04	464.64	465.00	7	0.11	454.50	465.00	8
A-n34-k5-C17	3	11.72	489	1.65	489.00	489.00	3	0.22	483.22	489.00	12
A-n36-k5-C18	3	18.05	502	13.92	483.95	502.00	171	3.49	474.83	502.00	178
A-n37-k5-C19	3	14.61	432	0.47	432.00	432.00	3	0.18	425.50	432	6
A-n37-k6-C19	3	12.48	584	10.87	569.77	584.00	100	13.52	548.83	584.00	530
A-n38-k5-C19	3	18.39	476	1.93	471.29	476.00	12	0.60	455.25	476.00	30
A-n39-k5-C20	3	14.89	557	6.91	532.05	557.00	64	15.10	521.43	557.00	605
A-n39-k6-C20	3	16.25	544	2.36	535.88	544.00	21	1.11	525.50	544.00	32
A-n44-k6-C22	3	17.61	608	7.43	595.53	608.00	79	9.16	563.88	608.00	229
A-n45-k6-C23	4	18.80	613	5.60	599.13	613.00	81	4.35	581.75	613.00	174
A-n45-k7-C23	4	23.09	674	373.30	637.25	674.00	2301	275.07	623.20	674.00	3435
A-n46-k7-C23	4	21.45	593	11.65	575.65	593.00	95	4.55	561.50	593.00	101
A-n48-k7-C24	4	20.62	667	197.96	631.04	667.00	1576	111.44	614.92	667.00	2471
A-n53-k7-C27	4	28.11	603	12.93	594.47	603.00	88	5.50	583.06	603.00	60
A-n54-k7-C27	4	31.43	690	51.62	670.97	690.00	243	52.35	655.86	690.00	631
A-n55-k9-C28	5	27.24	699	17.08	683.10	699.00	98	61.60	660.75	699.00	907
A-n60-k9-C30	5	31.98	769	40.83	753.67	769.00	202	30.98	738.21	769.00	298
A-n61-k9-C31	5	34.43	638	50.01	616.52	638.00	216	21.95	620.13	638.00	177
A-n62-k8-C31	4	44.84	740	39.15	717.39	740.00	113	37.44	712.42	740.00	287
A-n63-k10-C32	5	35.88	801	2772.27	767.65	801.00	7109	1840.60	753.17	801.00	8649
A-n63-k9-C32	5	47.81	912	5795.28	876.76	912.00	12872	6406.50	856.77	912	22655
A-n64-k9-C32	5	35.90	763	202.36	743.46	763.00	626	328.81	724.75	763.00	1900
A-n65-k9-C33	5	43.50	682	41.68	657.24	682.00	156	18.90	659.02	682.00	122
A-n69-k9-C35	5	39.60	680	338.26	652.38	680.00	1257	313.33	639.68	680.00	2157
A-n80-k10-C40	5	74.11	997	7200.07	942.54	969.44	7262	7200.90	902.63	946.14	10846
B-n31-k5-C16	3	9.85	441	0.47	441.00	441.00	5	0.07	440.00	441.00	6
B-n34-k5-C17	3	13.09	472	0.36	472.00	472.00	0	0.03	472.00	472.00	0
B-n35-k5-C18	3	14.64	626	0.79	625.30	626.00	9	0.12	623.50	626.00	6
B-n38-k6-C19	3	14.47	451	1.69	449.92	451.00	6	0.30	449.00	451.00	11
B-n39-k5-C20	3	18.76	357	0.81	354.08	357.00	7	0.25	353.50	357.00	12
B-n41-k6-C21	3	15.07	481	2.89	476.30	481.00	9	1.71	469.99	481.00	82
B-n43-k6-C22	3	23.46	483	11.80	472.94	483.00	100	4.54	471.30	483.00	134
B-n44-k7-C22	4	15.44	540	1.11	538.45	540.00	10	1.85	531.63	540	74
B-n45-k5-C23	3	26.53	497	1.37	497.00	497.00	1	0.67	491.83	497.00	13
B-n45-k6-C23	4	20.47	478	40.64	469.37	478.00	468	23.72	464.67	478.00	896
B-n50-k7-C25	4	24.72	449	1.97	449.00	449.00	1	1.37	441.00	449.00	42
B-n50-k8-C25	5	25.23	916	128.90	891.23	916.00	685	752.02	887.31	916.00	9005
B-n51-k7-C26	4	23.71	651	2.52	650.91	651.00	3	0.60	646.75	651.00	9
B-n52-k7-C26	4	24.37	450	0.96	450.00	450.00	3	0.30	446.50	450.00	4
B-n56-k7-C26	4	30.85	486	2.85	483.64	486.00	13	2.57	482.75	486.00	31
B-n57-k7-C29	4	31.09	751	2.32	751.00	751.00	0	4.60	745.91	751.00	95
B-n57-k9-C29	5	28.34	942	7.15	937.84	942.00	22	18.67	923.50	942.00	206
B-n63-k10-C32	5	39.01	816	23.21	802.34	816.00	118	18.70	795.50	816.00	183
B-n64-k9-C32	5	39.67	509	3.20	508.90	509.00	10	1.58	507.00	509.00	8
B-n66-k9-C33	5	45.72	808	20.31	799.19	808.00	48	9.01	798.79	808.00	37
B-n67-k10-C34	5	55.90	673	60.10	662.56	673.00	183	50.13	660.50	673.00	370
B-n68-k9-C34	5	43.95	704	11.23	701.77	704.00	14	7.19	698.33	704.00	42
B-n78-k10-C39	5	63.58	803	448.76	789.64	803.00	1066	220.81	785.79	803.00	933
P-n16-k8-C8	5	0.90	239	0.05	239.00	239.00	0	0.01	239.00	239.00	0
P-n19-k2-C10	2	3.30	147	0.04	147.00	147.00	0	0.00	147.00	147.00	0
P-n20-k2-C10	2	4.25	154	0.05	154.00	154.00	0	0.01	154.00	154.00	1
P-n21-k2-C11	2	4.94	160	0.08	160.00	160.00	1	0.02	156.75	160.00	5
P-n22-k2-C11	2	5.89	162	0.32	162.00	162.00	0	0.03	159.50	162.00	3
P-n22-k8-C11	5	2.40	314	0.71	314.00	314.00	0	0.04	311.50	314.00	6
P-n23-k8-C12	5	2.05	312	1.64	309.76	312.00	8	0.17	299.83	312.00	23
P-n40-k5-C20	3	19.88	294	2.40	286.91	294.00	28	1.34	279.58	294.00	69
P-n45-k5-C23	3	24.80	337	2.25	332.14	337.00	9	1.42	326.40	337.00	36
P-n50-k10-C25	5	21.81	410	108.92	391.58	410.00	647	197.47	373.13	410.00	2796
P-n50-k7-C25	4	31.38	353	13.26	342.84	353.00	102	19.30	335.10	353.00	444
P-n50-k8-C25	5	22.66	372	48.05	356.64	372.00	312	91.86	340.87	372.00	1329
P-n51-k10-C26	6	17.76	427	5.28	419.02	427.00	26	20.64	410.64	427.00	284
P-n55-k10-C28	5	26.61	415	187.84	397.27	415.00	960	406.26	381.50	415.00	4581
P-n55-k15-C28	9	18.10	551	1141.96	524.82	551.00	2650	2378.86	502.41	551.00	10911
P-n55-k7-C28	4	39.95	361	52.69	346.86	361.00	329	91.16	338.60	361.00	1372
P-n55-k8-C28	4	37.84	361	16.49	351.34	361.00	83	18.96	342.00	361.00	320
P-n60-k10-C30	5	35.88	443	7200.05	417.10	437.29	16120	7200.05	401.53	442.00	30780
P-n60-k15-C30	8	24.07	565	3441.63	539.53	565.00	6012	7200.06	517.53	559.13	25032
P-n65-k10-C33	5	39.89	487	340.68	471.59	487.00	756	531.44	454.80	487.00	3079
P-n70-k10-C35	5	48.51	485	126.43	474.24	485.00	307	123.25	462.75	485.00	614
P-n76-k4-C38	2	99.33	383	33.97	376.62	383.00	85	25.56	373.23	383	89
P-n76-k5-C38	3	90.45	405	16.16	397.79	405.00	24	26.76	394.56	405.00	89
P-n101-k4-C51	2	221.56	455	257.03	446.49	455.00	309	381.16	444.22	455.00	537

Table 3: Results for Bektaş instances with $\theta = 2$

	Metaheuristic			Ha et al.				Bektaş et al.			
Data	m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
A-n32-k5-C11	2	7.62	386	1.04	386.00	386.00	0	0.07	360.00	386.00	20
A-n33-k5-C11	2	8.62	315	3.55	314.93	315.00	0	0.08	302.00	315.00	14
A-n33-k6-C11	2	5.21	370	1.83	361.72	370.00	7	0.21	346.28	370	25
A-n34-k5-C12	2	9.12	419	3.93	413.22	419.00	9	0.57	389.45	419	76
A-n36-k5-C12	2	10.69	396	3.90	391.84	396.00	9	0.60	358.00	396.00	82
A-n37-k5-C13	2	11.55	347	0.67	347.00	347.00	2	0.12	339.11	347.00	6
A-n37-k6-C13	2	10.52	431	6.53	412.76	431.00	64	5.96	385.11	431.00	404
A-n38-k5-C13	2	12.27	367	2.24	366.03	367.00	2	0.12	352.00	367.00	14
A-n39-k5-C13	2	17.34	364	5.19	351.18	364.00	31	1.93	322.60	364.00	139
A-n39-k6-C13	2	9.18	403	2.97	402.89	403.00	2	0.16	386.46	403.00	5
A-n44-k6-C15	3	18.76	491	17.27	467.02	491.00	155	28.43	438.41	491.00	696
A-n45-k6-C15	3	13.64	474	4.34	464.03	474.00	9	0.77	437.62	474.00	53
A-n45-k7-C15	3	17.15	475	3.63	461.11	475.00	28	1.96	431.00	475.00	100
A-n46-k7-C16	3	19.68	462	10.77	438.01	462.00	106	7.14	416.65	462.00	355
A-n48-k7-C16	3	19.75	451	12.22	428.69	451.00	133	5.29	416.93	451.00	249
A-n53-k7-C18	3	31.21	440	5.00	421.68	440.00	37	3.58	411.73	440.00	134
A-n54-k7-C18	3	33.36	482	7.35	467.00	482.00	50	29.62	432.63	482.00	680
A-n55-k9-C19	3	23.60	473	8.21	461.57	473.00	71	3.30	446.92	473.00	100
A-n60-k9-C20	3	29.94	595	205.72	563.87	595.00	1192	351.78	532.55	595.00	4810
A-n61-k9-C21	4	28.13	473	11.40	452.12	473.00	68	6.83	439.18	473.00	171
A-n62-k8-C21	3	46.02	596	272.94	557.08	596.00	1133	336.91	542.60	596.00	3644
A-n63-k10-C21	4	24.56	593	306.24	560.99	593.00	1847	74.85	543.11	593.00	1383
A-n63-k9-C21	3	31.25	642	1077.89	608.97	642.00	4458	7200.05	570.31	624.33	29945
A-n64-k9-C22	3	44.78	536	30.83	513.20	536.00	137	12.17	511.86	536.00	97
A-n65-k9-C22	3	30.35	500	10.15	479.37	500.00	63	10.22	459.04	500.00	213
A-n69-k9-C23	3	36.56	520	521.66	487.13	520.00	2550	1664.10	461.66	520.00	13408
A-n80-k10-C27	4	85.11	710	7084.87	669.70	710.00	15132	7200.06	617.94	668.30	15774
B-n31-k5-C11	2	9.17	356	0.54	355.55	356.00	6	0.06	353.00	356.00	14
B-n34-k5-C12	2	10.90	369	0.12	369.00	369.00	0	0.02	368.33	369.00	5
B-n35-k5-C12	2	12.46	501	1.19	500.08	501.00	0	0.08	496.50	501.00	11
B-n38-k6-C13	2	13.01	370	2.54	366.91	370.00	9	0.41	360.97	370.00	28
B-n39-k5-C13	2	8.17	280	0.25	280.00	280.00	3	0.03	279.00	280.00	2
B-n41-k6-C14	2	10.90	407	1.25	407.00	407.00	0	0.28	396.00	407.00	19
B-n43-k6-C15	2	17.99	343	1.59	342.33	343.00	3	0.32	338.00	343.00	12
B-n44-k7-C15	3	12.15	395	3.45	388.14	395.00	17	0.93	383.00	395.00	120
B-n45-k5-C15	2	22.52	410	1.80	409.89	410.00	2	0.20	404.00	410.00	17
B-n45-k6-C15	2	17.57	336	2.00	335.05	336.00	8	0.76	328.75	336.00	40
B-n50-k7-C17	3	25.56	393	1.65	392.98	393.00	2	0.27	390.00	393.00	10
B-n50-k8-C17	3	19.36	598	19.80	589.09	598.00	167	14.41	572.75	598.00	735
B-n51-k7-C17	3	18.90	511	2.02	510.37	511.00	2	0.26	506.50	511.00	9
B-n52-k7-C18	3	18.31	359	0.30	359.00	359.00	0	0.04	359.00	359.00	0
B-n56-k7-C19	3	27.55	356	4.13	351.50	356.00	11	15.70	339.00	356.00	807
B-n57-k7-C19	3	27.69	558	1.73	557.84	558.00	1	1.26	554.00	558.00	27
B-n57-k9-C19	3	29.93	681	280.72	665.25	681.00	1816	201.81	657.74	681.00	3762
B-n63-k10-C21	3	44.98	599	5.96	593.93	599.00	20	6.00	584.50	599.00	119
B-n64-k9-C22	4	34.85	452	3.36	449.05	452.00	16	1.58	445.00	452.00	28
B-n66-k9-C22	3	28.40	609	179.18	586.17	609.00	1180	44.74	583.57	609.00	1129
B-n67-k10-C23	4	54.13	558	9.89	548.63	558.00	40	21.09	451.00	558.00	461
B-n68-k9-C23	3	31.58	523	43.00	514.79	523.00	259	41.59	504.50	523.00	1034
B-n78-k10-C26	4	59.67	606	8.32	601.66	606.00	23	3.90	596.20	606.00	26
P-n16-k8-C6	4	1.20	170	0.03	170.00	170.00	0	0.00	170.00	170.00	0
P-n19-k2-C7	1	3.01	111	0.08	111.00	111.00	0	0.01	106.50	111.00	3
P-n20-k2-C7	1	2.55	117	0.49	117.00	117.00	1	0.02	111.90	117.00	6
P-n21-k2-C7	1	3.11	117	0.19	117.00	117.00	0	0.03	115.06	117	4
P-n22-k2-C8	1	3.15	111	0.08	111.00	111.00	0	0.01	111.00	111.00	0
P-n22-k8-C8	4	1.45	249	0.19	244.89	249.00	11	0.05	233.50	249.00	18
P-n23-k8-C8	3	1.66	174	0.11	174.00	174.00	2	0.01	174.00	174.00	0
P-n40-k5-C14	2	18.09	213	3.68	208.92	213.00	9	0.20	203.75	213.00	16
P-n45-k5-C15	2	22.08	238	4.34	225.63	338.00	47	3.56	209.29	238.00	244
P-n50-k10-C17	4	19.40	292	2.03	287.57	292.00	7	1.83	276.17	292.00	49
P-n50-k7-C17	3	23.62	261	3.78	249.97	261.00	27	2.60	237.84	261.00	130
P-n50-k8-C17	3	23.62	262	1.77	258.76	262.00	8	1.39	248.30	262.00	42
P-n51-k10-C17	4	24.59	309	25.61	292.43	309.00	216	36.71	264.38	309.00	1457
P-n55-k10-C19	4	30.75	301	5.94	292.99	301.00	38	8.00	271.18	301.00	219
P-n55-k15-C19	6	16.15	378	8.59	366.22	378.00	41	12.21	343.37	378.00	244
P-n55-k7-C19	3	46.62	271	20.76	255.88	271.00	162	25.64	236.11	271.00	742
P-n55-k8-C19	3	47.46	274	27.64	261.01	274.00	193	23.57	242.20	274.00	683
P-n60-k10-C20	4	34.38	325	41.05	306.24	325.00	296	70.48	287.50	325.00	1870
P-n60-k15-C20	6	23.57	374	213.65	352.20	374.00	868	302.16	333.28	374.00	3692
P-n65-k10-C22	4	43.86	372	41.62	354.30	372.00	199	439.28	328.78	372.00	6155
P-n70-k10-C24	4	57.58	385	514.15	365.59	385.00	1970	1144.44	343.61	385.00	12857
P-n76-k4-C26	2	126.48	309	72.86	294.79	309.00	214	64.03	281.00	309.00	655
P-n76-k5-C26	2	126.82	309	21.39	301.85	309.00	90	59.48	281.00	309.00	541
P-n101-k4-C34	2	294.95	370	1448.16	352.99	370.00	3065	1247.68	347.71	370.00	4073

Table 4: Results for Bektaş instances with $\theta = 3$

Data	θ	Metaheuristic			Ha et al.				Bektaş et al.			
		m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
M-n101-k10-C51	2	5	124.23	542	519.56	534.40	542.00	714	2454.36	527.00	542.00	4834
M-n121-k7-C61	2	4	234.26	719	21600.30	650.88	705.84	4653	21600.15	683.60	705.25	10677
M-n151-k12-C76	2	6	305.96	659	21600.26	626.88	634.65	3238	21600.30	612.10	623.28	4226
M-n200-k16-C100	2	9	453.95	789	21601.15	746.30	752.14	892	21600.85	726.58	734.08	1386
G-n262-k25-C131	2	13	822.84	3303	21610.65	2930.49	2945.02	233	21603.67	2809.10	2821.84	328
M-n101-k10-C34	3	4	152.31	458	2831.89	442.82	458.00	7781	5903.39	433.99	458.00	3882
M-n121-k7-C41	3	3	238.23	527	1639.85	513.85	527.00	2001	1669.98	504.18	527.00	3001
M-n151-k12-C51	3	4	434.29	483	21600.29	462.73	474.32	7865	21600.20	438.45	459.06	10689
M-n200-k16-C67	3	6	601.99	605	21600.56	564.57	572.81	2127	21600.46	543.34	554.54	3720
G-n262-k25-C88	3	9	861.73	2477	21601.39	2227.79	2239.50	508	21601.36	2035.83	2053.25	785

Table 5: Results for large Bektaş instances

	Ha et al.					Bektaş et al.			
Data	m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
A-n32-k5-C16	3	508	151	401	844	3	508	1228	211
A-n33-k5-C17	3	451	99	230	404	3	451	617	75
A-n33-k6-C17	3	465	44	73	223	3	465	620	31
A-n34-k5-C17	3	489	58	130	269	3	489	636	56
A-n36-k5-C18	3	502	165	417	891	3	502	1109	180
A-n37-k5-C19	3	432	27	49	248	3	432	718	46
A-n37-k6-C19	3	584	135	400	986	3	584	2188	257
A-n38-k5-C19	3	476	103	184	479	3	476	808	83
A-n39-k5-C20	3	557	74	262	969	3	557	2451	270
A-n39-k6-C20	3	544	113	305	910	3	544	1715	166
A-n44-k6-C22	3	608	119	326	1143	3	608	2487	232
A-n45-k6-C23	4	613	77	238	769	4	613	1653	112
A-n45-k7-C23	4	674	228	719	3818	4	674	7249	433
A-n46-k7-C23	4	593	128	306	1042	4	593	2696	133
A-n48-k7-C24	4	667	200	619	2836	4	667	5075	264
A-n53-k7-C27	4	603	155	400	1265	4	603	3094	209
A-n54-k7-C27	4	690	222	581	2057	4	690	4772	357
A-n55-k9-C28	5	699	143	390	1546	5	699	4982	316
A-n60-k9-C30	5	769	181	566	2071	5	769	5400	288
A-n61-k9-C31	5	638	171	407	2034	5	638	5901	216
A-n62-k8-C31	4	740	231	555	2855	4	740	5915	359
A-n63-k10-C32	5	801	402	1157	7363	5	801	13364	634
A-n63-k9-C32	5	912	512	1425	7337	5	912	15093	995
A-n64-k9-C32	5	763	268	795	3470	5	763	8165	564
A-n65-k9-C33	5	682	154	432	2861	5	682	5730	196
A-n69-k9-C35	5	680	217	661	4775	5	680	9105	303
A-n80-k10-C40	-	-	595	1668	10444	-	-	22189	1287
B-n31-k5-C16	3	441	25	56	146	3	441	447	31
B-n34-k5-C17	3	472	25	49	142	3	472	307	22
B-n35-k5-C18	3	626	48	61	150	3	626	527	38
B-n38-k6-C19	3	451	43	120	383	3	451	914	41
B-n39-k5-C20	3	357	39	92	211	3	357	605	35
B-n41-k6-C21	3	481	44	224	724	3	481	1821	114
B-n43-k6-C22	3	483	101	325	1164	3	483	2779	165
B-n44-k7-C22	4	540	65	175	548	4	540	1645	122
B-n45-k5-C23	3	497	49	107	490	3	497	701	70
B-n45-k6-C23	4	478	134	440	1375	4	478	3027	253
B-n50-k7-C25	4	449	49	95	253	4	449	667	60
B-n50-k8-C25	5	916	264	769	1852	5	916	9775	934
B-n51-k7-C26	4	651	36	84	751	4	651	1113	43
B-n52-k7-C26	4	450	29	46	303	4	450	1014	31
B-n56-k7-C26	4	486	80	205	720	4	486	3131	105
B-n57-k7-C29	4	751	54	155	913	4	751	3925	110
B-n57-k9-C29	5	942	202	423	1316	5	942	4794	319
B-n63-k10-C32	5	816	135	373	1450	5	816	3749	194
B-n64-k9-C32	5	509	56	129	729	5	509	1953	72
B-n66-k9-C33	5	808	230	540	3458	5	808	7043	204
B-n67-k10-C34	5	673	175	436	2362	5	673	7995	317
B-n68-k9-C34	5	704	281	690	2196	5	704	5463	218
B-n78-k10-C39	5	803	352	919	5290	5	803	12921	478
P-n16-k8-C8	5	239	10	17	39	5	239	94	12
P-n19-k2-C10	2	147	8	13	29	2	147	55	14
P-n20-k2-C10	2	154	9	17	42	2	154	43	18
P-n21-k2-C11	2	160	9	11	26	2	160	89	22
P-n22-k2-C11	2	162	19	45	61	2	162	119	33
P-n22-k8-C11	5	314	40	88	143	5	314	303	29
P-n23-k8-C12	5	312	21	160	291	5	312	571	45
P-n40-k5-C20	3	294	98	171	411	3	294	935	106
P-n45-k5-C23	3	337	100	193	588	3	337	1123	94
P-n50-k10-C25	5	410	178	460	2502	5	410	5334	377
P-n50-k7-C25	4	353	179	369	938	4	353	3263	317
P-n50-k8-C25	5	372	163	445	2439	5	372	5049	363
P-n51-k10-C26	6	427	111	260	1094	6	427	3660	267
P-n55-k10-C28	5	415	215	519	3141	5	415	6091	400
P-n55-k15-C28	9	551	328	1008	6528	9	551	16484	533
P-n55-k7-C28	4	361	151	356	1603	4	361	4656	292
P-n55-k8-C28	4	361	150	297	1160	4	361	2441	174
P-n60-k10-C30	-	-	472	1327	11477	-	-	19820	796
P-n60-k15-C30	8	565	287	991	10402	-	-	24928	641
P-n65-k10-C33	5	487	251	612	3915	5	487	9052	473
P-n70-k10-C35	5	485	237	508	3020	5	485	8889	328
P-n76-k4-C38	2	383	182	417	1709	2	238	4507	284
P-n76-k5-C38	3	405	183	380	1372	3	405	4893	244
P-n101-k4-C51	2	455	319	735	3490	2	455	9470	586

Table 6: Details of branch-and-cut algorithms for Bektaş instances with $\theta = 2$

	Ha et al.					Bektaş et al.			
Data	m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
A-n32-k5-C11	2	386	61	60	168	2	386	195	43
A-n33-k5-C11	2	315	92	151	178	2	315	188	43
A-n33-k6-C11	2	370	59	120	313	2	370	351	68
A-n34-k5-C12	2	419	83	208	209	2	419	350	86
A-n36-k5-C12	2	396	126	237	146	2	396	187	114
A-n37-k5-C13	2	347	45	71	125	2	347	234	65
A-n37-k6-C13	2	431	134	331	381	2	431	679	306
A-n38-k5-C13	2	367	45	70	162	2	367	284	51
A-n39-k5-C13	2	364	154	269	436	2	364	461	220
A-n39-k6-C13	2	403	93	144	269	2	403	449	67
A-n44-k6-C15	3	491	199	429	862	3	491	1096	659
A-n45-k6-C15	3	474	112	131	465	3	474	424	116
A-n45-k7-C15	3	475	113	268	344	3	475	685	198
A-n46-k7-C16	3	462	168	327	453	3	462	819	283
A-n48-k7-C16	3	451	176	387	698	3	451	860	254
A-n53-k7-C18	3	440	123	273	482	3	440	836	218
A-n54-k7-C18	3	482	151	333	581	3	482	1195	541
A-n55-k9-C19	3	473	128	283	666	3	473	1364	168
A-n60-k9-C20	3	595	336	813	1601	3	595	3054	1016
A-n61-k9-C21	4	473	144	292	760	4	473	1429	213
A-n62-k8-C21	3	596	382	892	1226	3	596	2637	823
A-n63-k10-C21	4	593	347	781	1921	4	593	2715	456
A-n63-k9-C21	3	642	486	1238	2700	3	642	5855	2114
A-n64-k9-C22	3	536	253	546	938	3	536	2415	739
A-n65-k9-C22	3	500	181	369	741	3	500	1732	214
A-n69-k9-C23	3	520	383	848	2326	3	520	6151	948
A-n80-k10-C27	4	710	846	1737	3996	-	-	6418	3198
B-n31-k5-C11	2	356	50	76	68	2	356	175	42
B-n34-k5-C12	2	369	23	27	63	2	369	97	23
B-n35-k5-C12	2	501	47	68	63	2	501	111	49
B-n38-k6-C13	2	370	75	153	288	2	370	515	130
B-n39-k5-C13	2	280	21	32	60	2	280	192	29
B-n41-k6-C14	2	407	47	80	143	2	407	342	104
B-n43-k6-C15	2	343	71	108	110	2	343	393	75
B-n44-k7-C15	3	395	75	186	300	3	395	459	90
B-n45-k5-C15	2	410	63	73	124	2	410	219	49
B-n45-k6-C15	2	336	79	169	228	2	336	583	138
B-n50-k7-C17	3	393	74	88	297	3	393	468	53
B-n50-k8-C17	3	598	190	494	613	3	598	1299	369
B-n51-k7-C17	3	511	55	96	267	3	511	439	64
B-n52-k7-C18	3	359	35	38	205	3	359	450	27
B-n56-k7-C19	3	356	121	316	219	3	356	703	327
B-n57-k7-C19	3	558	52	90	315	3	558	710	200
B-n57-k9-C19	3	681	321	787	911	3	681	2077	698
B-n63-k10-C21	3	599	227	431	792	3	599	1206	205
B-n64-k9-C22	4	452	106	217	240	4	452	1033	118
B-n66-k9-C22	3	609	284	768	2014	3	609	3380	383
B-n67-k10-C23	4	558	167	308	379	4	558	1338	233
B-n68-k9-C23	3	523	212	703	1171	3	523	1772	298
B-n78-k10-C26	4	606	179	348	1032	4	606	2084	149
P-n16-k8-C6	4	170	4	14	26	4	170	60	6
P-n19-k2-C7	1	111	13	15	20	1	111	23	18
P-n20-k2-C7	1	117	21	44	27	1	117	44	35
P-n21-k2-C7	1	117	23	26	19	1	117	83	46
P-n22-k2-C8	1	111	14	25	20	1	111	41	18
P-n22-k8-C8	4	249	42	135	87	4	249	184	32
P-n23-k8-C8	3	174	10	22	41	3	174	100	13
P-n40-k5-C14	2	213	101	146	231	2	213	403	75
P-n45-k5-C15	2	238	171	292	421	2	238	686	224
P-n50-k10-C17	4	292	128	195	403	4	292	907	196
P-n50-k7-C17	3	261	173	260	313	3	261	660	179
P-n50-k8-C17	3	262	123	184	387	3	262	872	175
P-n51-k10-C17	4	309	189	372	659	4	309	1468	350
P-n55-k10-C19	4	301	194	285	554	4	301	1275	237
P-n55-k15-C19	6	378	180	345	1044	6	378	2135	272
P-n55-k7-C19	3	271	195	342	642	3	271	1125	340
P-n55-k8-C19	3	274	196	356	652	3	274	1125	358
P-n60-k10-C20	4	325	253	457	1087	4	325	1999	403
P-n60-k15-C20	6	374	307	699	1601	6	374	3965	735
P-n65-k10-C22	4	372	269	446	875	4	372	3303	649
P-n70-k10-C24	4	385	406	788	1658	4	385	3619	977
P-n76-k4-C26	2	309	228	368	1167	2	309	1693	467
P-n76-k5-C26	2	309	190	287	860	2	309	2576	484
P-n101-k4-C34	2	370	630	1252	1860	2	370	3265	1385

Table 7: Details of branch-and-cut algorithms for Bektaş instances with $\theta = 3$

		Ha et al.					Bektaş et al.			
Data	θ	m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
M-n101-k10-C51	2	5	542	334	761	4569	5	542	15370	664
M-n121-k7-C61	2	-	-	756	2143	25240	-	-	59265	1510
M-n151-k12-C76	2	-	-	801	1863	30410	-	-	79106	1659
M-n200-k16-C100	2	-	-	786	2324	41990	-	-	129218	1774
G-n262-k25-C131	2	-	-	911	2744	38569	-	-	286211	2236
M-n101-k10-C34	3	4	458	563	1268	1921	4	458	3882	1407
M-n121-k7-C41	3	3	527	693	1473	7741	3	527	15872	1145
M-n151-k12-C51	3	-	-	992	2008	12557	-	-	22870	2373
M-n200-k16-C67	3	-	-	1171	2319	13683	-	-	39806	3141
G-n262-k25-C88	3	-	-	1243	3097	22258	-	-	93227	2475

Table 8: Details of branch-and-cut algorithms for large Bektaş instances