CIRRELT

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 Multi-Vehicle Covering Tour Problem with a Constraint on the Number of Vertices

Minh Hoang Ha
Nathalie Bostel
André Langevin
Louis-Martin Rousseau

November 2012

CIRRELT-2012-64

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

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

An Exact Algorithm and a Metaheuristic for the Multi-Vehicle Covering Tour Problem with a Constraint on the Number of Vertices

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 multi-vehicle covering tour problem (*m*-CTP) involves finding a minimum length set of vehicle routes passing through a subset of vertices, subject to constraints on the length of each route and the number of vertices that it contains, such that each vertex not included in any route lies within a given distance of a route. This paper tackles a particular case of *m*-CTP where only the restriction on the number of vertices is considered, i.e., the constraint on the length is relaxed. The problem is solved by a branch-and-cut algorithm and a metaheuristic. To develop the branch-and-cut algorithm, we use a new integer programming formulation based on a two-commodity flow model. The metaheuristic is based on the evolutionary local search (ELS) method proposed in [23]. Computational results are reported for a set of test problems derived from the TSPLIB.

Keywords. Covering tour, 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: Andre.Langevin@cirrelt.ca

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

1. Introduction

The *multi-vehicle covering tour problem* (m-CTP) is a generalization of the *vehicle routing problem* (VRP), which is an extension of the *covering tour problem* (CTP). The m-CTP is defined as follows.

Let $G = (V \cup W, E_1 \cup E_2)$ be an undirected graph, where $V \cup W$ is the vertex set and $E_1 \cup E_2$ is the edge set. $V = \{v_0, ..., v_{n-1}\}$ is the set of n vertices that can be visited, and $W = \{w_1, w_2, ..., w_l\}$ is the set of vertices that must be covered. Let $T = \{v_0, ..., v_{t-1}\}$, a subset of V, be the set of vertices that must be visited. Vertex v_0 is the depot; m identical vehicles are located there. This paper considers the case where m is a decision variable. A length c_{ij} is associated with each edge of $E_1 = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ and a distance d_{ij} is associated with each edge of $E_2 = \{(v_i, v_j) : v_i \in V \setminus T, v_j \in W\}$. The m-CTP consists in finding m vehicle routes such that the total cost is minimized and

- Each route begins and ends at the depot;
- Each vertex of T is visited exactly once while each vertex of $V \setminus T$ is visited at most once;
- Each vertex j of W is covered by the routes, i.e., lies within a distance r of at least one vertex of $V \setminus T$ that is visited, where r is the covering radius;
- The number of vertices on each route (excluding the depot) is less than a given value p;
- The length of each route does not exceed a fixed limit q.

The m-CTP is clearly NP-hard since it reduces to a VRP with unit demands when$T = V$ and $W = \emptyset$ or to a CTP when the capacity constraints are relaxed.

The m-CTP can model problems concerned with the design of bilevel transportation networks, such as the construction of routes for mobile health-care teams (see, e.g., [15, 28]) and mobile library teams, and the location of post boxes [18], banking agencies, and milk collection points [27]. Recently, the model of the CTP has also been used to solve the disaster relief problem in [8]. In this application, after a disaster the health care organizations have to supply the affected populations with food, water and medicine. The relief vehicles (e.g. mobile hospitals) stop at several locations and the populations (the set W in the mathematical description of the problem) must visit one of the vehicle stops. The health care organizations have to choose the appropriate stops among $|V|$ potential locations so that all populations can reach one of these stops within acceptable time. T can be considered as the set of stops covering the populations that cannot be covered by other stops.

Many researchers have studied classical vehicle routing problems where all the customers have to be served, but the number of papers on the CTP is much more limited. It seems that the first work on this problem can be credited to [7] in 1981. Since then the CTP has received little attention from the research community. The one-vehicle version (1-CTP) was solved exactly by a branch-and-cut algorithm in [10]. A heuristic was also proposed in this paper. The authors of [3] used a two-commodity flow formulation and developed a scatter search algorithm. For the multi-vehicle version, [14] introduced a three-index vehicle flow formulation and three heuristics inspired by classical algorithms: Clarke and Wright [6], the sweep algorithm [11], and the route-first/cluster-second method [5]. The three heuristics are compared to each other, and the optimality gap is therefore unknown. Recently, [16] has proposed the first exact algorithm. It is based on a column generation approach in which the master problem is a simple set covering problem, and the subproblem is formulated similarly to the 1-CTP model of [10]. The algorithm was tested on instances derived from TSPLIB, and results for $|V|+|W|=100$ and $|T|=1$ are reported.

The close-enough arc routing problem (CEARP) can be seen as an arc-routing counterpart of CTP. It is similar to the CTP except that a closed tour must be determined so that every vertex of W lies within a distance r of an *arc* of the tour. This problem was considered in [12, 13].

In this paper, we address a particular case of m-CTP where the length constraint is relaxed, i.e., $q=+\infty$. We refer to this as m-CTP-p (p is the upper bound on the number of vertices per route). This version can model the applications where the distance constraint is not important and can be relaxed. An example is in the vaccination campaigns where the health care team has to vaccinate the populations at several locations. When the travelling time among the locations is quite small compared with the service duration at a location, we can consider that each vehicle can serve only a limited number of locations and the distance constraint in this case can be neglected. Our contributions are that we present a new formulation for m-CTP-p and propose an exact method for this problem, as well as a metaheuristic. Computational experiments show that our exact approach outperforms the column generation method of [16], and our metaheuristic gives high-quality solutions for the tested instances.

The remainder of the paper is organized as follows. Section 2 describes our formulation and several valid inequalities. The branch-and-cut algorithm and 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 m-CTP-p

In this section, we describe a new integer programming formulation for m-CTP-p. The idea underlying this formulation was first introduced by [9] for the traveling salesman problem (TSP). Langevin et al. [19] extended this approach to solve the TSP with time windows. Baldacci et al. [4] used this method to derive a new formulation and a branch-and-cut for the VRP, and Baldacci et al. [3] adapted it to formulate the 1-CTP without the capacity constraints.

Our formulation is an extension of that proposed by Baldacci et al. [3] for the 1-CTP. To adapt this idea for m-CTP-p, we consider that each vertex of $V \setminus \{v_0\}$ has a unit demand and each vehicle has a capacity of p. We also note that the difference between m-CTP-p and VRP is that we do not need to visit all vertices of V with the exception of the vertices of T.

The original graph G is first extended to $\overline{G} = (\overline{V} \cup W, \bar{E_1} \cup E_2)$ by adding a new vertex v_n, which is a copy of the depot v_0. We have $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E_1 \cup \{(v_i, v_n), v_i \in V'\}$, 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 m-CTP-p solution along which the vehicle initially carries a load of p units. When a vehicle travels from v_i to v_j, flow f_{ij} represents the number of vertices that can still be visited and flow f_{ji} represents the number of vertices already visited (i.e., $f_{ji} = p$ - 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 presence of vertex v_i in the solution. We set the binary coefficients λ_{il} equal to 1 if and only if $w_l \in W$ can be covered by $v_i \in V \setminus T$. Then m-CTP-p can be stated as:

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

$$\text{subject to} \quad \sum_{v_i\in V\setminus T} \lambda_{il}y_i \geq 1 \qquad \forall w_l \in W \tag{2}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{3}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2y_i \qquad \forall v_i \in V' \tag{4}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} y_i \tag{5}$$

$$\sum_{j\in V'} f_{nj} = mp \tag{6}$$

$$f_{ij} + f_{ji} = px_{ij} \qquad \forall \{v_i,v_j\} \in \overline{E} \tag{7}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall \{v_i,v_j\} \in \overline{E} \tag{8}$$

$$y_i = 1 \qquad \forall v_i \in T \setminus \{v_0\} \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall \{v_i,v_j\} \in \overline{E} \tag{10}$$

$$y_i \in \{0,1\} \qquad \forall v_i \in V' \tag{11}$$

$$m \in \mathbb{N}. \tag{12}$$

The objective (1) is to minimize the total travel cost. Constraints (2) ensure that every customer of W is covered, while constraints (3) ensure that each vertex of V' is visited at most once. Constraints (4) to (7) define the flow variables. Specifically, constraints (4) state that the inflow minus the outflow at each vertex $v_i \in V'$ is equal to 2 if v_i is used and to 0 otherwise. The outflow at the source vertex v_0 (5) is equal to the total demand of the vertices that are used in the solution, and the outflow at the sink v_n (6) corresponds to the total capacity of the vehicle fleet. Constraint (7) is derived from the definition of the flow variables. Constraints (10) and (12) define the variables.

Figure 1 shows a feasible solution of m-CTP-p with two routes in the case where $p = 3$ under the two-commodity form. The solid lines in the figure represent the flows f_{ij} while the dotted lines represent the flows f_{ji}.

A disadvantage of this formulation is that it can not express the constraint

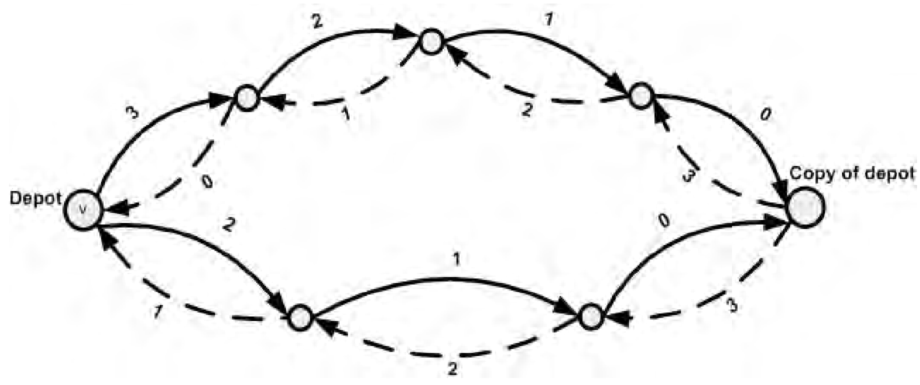

Figure 1: Flow paths for solution with two routes and p=3

on the length of each route. However, its advantages are that the number of variables and constraints increases polynomially with the size of the problem, and its LP relaxation satisfies a weak form of the subtour elimination constraints (see [4]).

The linear relaxation of m-CTP-p can be strengthened by the addition of valid inequalities. The valid inequalities for 1-CTP apply directly to our problem (see [10]). In the following dominance inequalities (14), a vertex v_i is said to dominate v_j if v_i can cover all the vertices of W that v_j can cover. In the dominance inequalities (15), this dominance relation is extended to three vertices where a subset of two vertices dominates the third vertex. We have

$$x_{ij} \leqslant y_i \text{ and } x_{ij} \leqslant y_j \; (v_i \text{ or } v_j \in V \setminus T) \tag{13}$$

$$y_i + y_j \leqslant 1 \text{ if } v_i \text{ dominates } v_j \text{ or conversely } (v_i, v_j \in V \setminus T) \tag{14}$$

$$\begin{aligned} y_i + y_j + y_k \leqslant 2 \text{ if two of } v_i, v_j, v_k \\ \text{dominate the other vertex } (v_i, v_j, v_k \in V \setminus T). \end{aligned} \tag{15}$$

All the valid inequalities of the set covering polytope $conv\{y : \sum b_a.y_a \geq 1,\ y_a \in \{0,1\}\}$ where b_a is the binary coefficient, are valid for m-CTP-p. Balas and Ng [2] introduced the facets with coefficients in {0,1,2} and Sánchez-García et al. [26] introduced the more complex facets with coefficients in {0,1,2,3}. Here, we recall the first one that was used in [10]: let S be a nonempty subset of W, and define for each $v_k \in V$ the coefficient

$$\alpha_k^S = \begin{cases} 0 \text{ if } \lambda_{kl} = 0 \text{ for all } w_l \in S, \\ 2 \text{ if } \lambda_{kl} = 1 \text{ for all } w_l \in S, \\ 1 \text{ otherwise.} \end{cases}$$

Then the following constraint is valid for m-CTP-p:

$$\sum_{v_k \in V} \alpha_k^S y_k \geq 2. \quad (16)$$

The following flow inequalities were introduced in [3]:

$$f_{ij} \geq x_{ij}, f_{ji} \geq x_{ji} \text{ if } i, j \neq v_0 \text{ and } i, j \neq v_n. \quad (17)$$

Several other valid inequalities for the VRP expressed for the set T of required vertices can be applied directly to our problem. Here we restrict ourselves to the capacity constraints, originally proposed by [20]:

$$\sum_{(v_i, v_j \in S)} x_{ij} \leq |S| - \left\lceil \frac{|S|}{p} \right\rceil (S \subseteq T, |S| \geq 2). \quad (18)$$

Let z be the minimum number of vertices required to cover all vertices of W. The following constraint follows immediately:

$$m \geq \left\lceil \frac{z}{p} \right\rceil. \quad (19)$$

The value of z can be calculated by solving a set covering problem as follows:

$$\text{Minimize} \qquad z = |T| + \sum_{(v_i \in V \setminus T)} y_i \quad (20)$$

$$\text{subject to} \qquad \sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1 \qquad \forall w_l \in W \quad (21)$$

$$y_i = 0, 1 \qquad \forall v_i \in V \setminus T. \quad (22)$$

3. Branch-and-cut algorithm

We solve m-CTP-p exactly using a standard branch-and-cut algorithm. We solve a linear program containing the constraints (1), (2), (3), (4), (5), (6), (7), (8), (9), and (19). We then search for violated constraints of type (13), (14), (15), (16), (17), and (18), and the detected constraints are added 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. If all the variables are integer, we explore another node.

The separation of the constraints of type (13), (14), (15), and (17) is straightforward. For constraints (16), as in [10], to reduce the computational effort we verify only the sets S that include three elements.

To generate the capacity constraints (18), we use the *greedy randomized algorithm*, proposed by [1] and reused in [4]. This is an iterative procedure that is applied to subsets $T' \subset T$ created a priori. At each iteration, the following procedure is repeated for each $S \in T'$. Let $v_{i^*} \in T \setminus S$ be a vertex such that

$$\sum_{j \in S} (x_{i^*j} + x_{ji^*}) = \max_{i \in T \setminus S} \left[\sum_{j \in S} (x_{ij} + x_{ji}) \right].$$

If the current solution x violates the capacity constraints (18) corresponding to the subset $S' = S \cup i^*$, then we add this inequality to the model, update S to S', and repeat the process until S contains all vertices of T. In our implementation, the initial set T' can be a single vertex of T because the number of vertices is fairly small in m-CTP-p instances.

Our branch-and-cut algorithm is built around CPLEX 11.2 with the Callable Library. We tested the algorithm with the activation of each CPLEX cut one by one, and observed that only two of them (Gomory cut and implied-bound cut) were useful. Thus, all CPLEX cuts except the Gomory and implied-bound cuts are turned off. Gomory cuts are generated by applying integer rounding on a pivot row in the optimal LP tableau for a (basic) integer variable with a fractional solution value. These cuts are applied to solve the VRP (see [21] for example). Implied-bound cuts are generated in some models where binary variables imply bounds on continuous variables. Unfortunately, we do not know why the implied-bound cuts are useful. All the other CPLEX parameters are set to their default values.

We tested several branching techniques, such as branching on the variables y before x as in [10] 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 introduce a metaheuristic for m-CTP-p that is a two-phase hybrid algorithm. The aim of the first phase is to randomly generate nt subsets of V such that each subset can cover all the customers. The vertices of each subset combined with T create a set N of the vertices that must be visited. The problem now becomes a VRP with unit demands, and it is solved in the second phase by an algorithm based on the ELS method of [23].

4.1. First phase

To randomly generate subsets of vertices covering all vertices of W, we solve the following θ_1 mixed integer programming problems:

$$\text{Minimize} \quad \sum_{v_i \in V \setminus T} b_i y_i \tag{23}$$

$$\text{subject to} \quad \sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1 \qquad \forall w_l \in W \tag{24}$$

$$y_i = 0, 1 \qquad \forall v_i \in V \setminus T \tag{25}$$

where b_i is a random number in $\{1,2\}$. The solution of this model is rapid even for the large instances in our tests.

4.2. Second phase

The goal of the second phase is to solve the VRP problems. We apply the ELS method proposed in [23] because of its simplicity, speed, and good performance. 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 introduce the procedures for the construction of the second phase.

4.2.1. Split, concat, and mutate procedures

The split procedure is the backbone of our metaheuristic. Its goal is to split a giant tour into VRP routes. This procedure was originally introduced in [5] and then integrated into memetic algorithms to successfully solve various vehicle routing problems (see [17, 24, 22] for example). The reader is referred to [25] for an efficient implementation of this procedure. The concat procedure does the opposite: it concatenates VRP routes into a giant tour.

The output of the mutate(L) procedure is a randomly perturbed copy L' of the input giant tour L. This procedure randomly swaps the position of two vertices in the original tour.

4.2.2. Local search procedures

As in [23], we can use classical moves such as 2-opt moves, Or-opt moves, or string-exchange moves. Here, we use two simple classical local searches: relocation of a node (LS2) and two-point moves that swap the position of two nodes (LS3). We also introduce two new approaches: saturation moves (LS1) that combine two unsaturated routes, and new-node moves (LS4) that try to replace a node in the solution by a new node.

Local search LS1. After we split the giant tour, some routes may not be saturated, and they can be combined with other routes if the total number of nodes in the two routes is less than p. In other words, this technique helps us to reduce the number of routes. To combine two routes (if possible), we consider the four methods illustrated in Fig. 2 and choose the best.

Local search LS4. This technique replaces a node in the tour by a new node that is not present in the current solution. The swap is done if it does not violate the covering constraint and improves the solution (see Fig. 3). LS4 is called after each ELS phase. If it can improve the solution, the new solution (i.e., after LS4) becomes the initial solution for a new ELS phase; otherwise, we return to the first phase.

After many tests, we decided to use *first-improvement local search* for LS1 and LS4 and *best-improvement local search* for LS2 and LS3. In first-improvement local search, if an improving move is detected, it is immediately executed and the remaining moves are bypassed. The process is repeated until we can not find a better solution. Best-improvement local search evaluates all the possible moves and executes the best one.

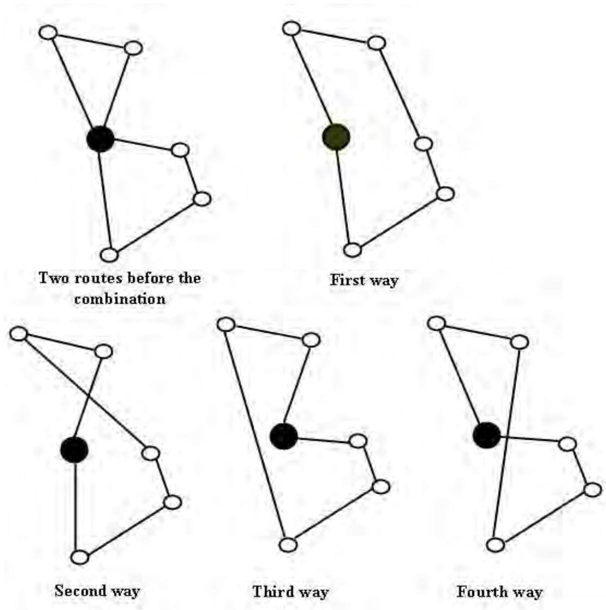

Figure 2: Four ways to combine two routes in LS1

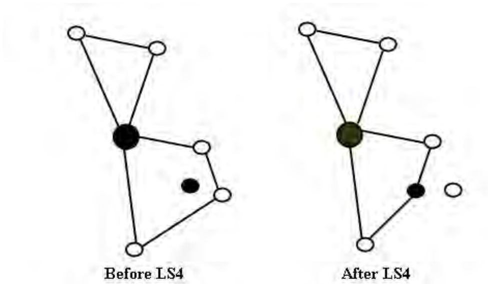

Figure 3: Local search LS4

4.2.3. Initial solution

We use the route-first/cluster-second approach to generate the initial solution. To create giant tours, we use two procedures: insertion of the nearest neighbor (Insert1) and the greatest-saving insertion (Insert2). We split these two tours to obtain two solutions. The initial solution is the best found by local search LS1.

4.2.4. Tabu list

In our algorithm, the tabu list stores attributes of the giant tours instead of final solutions. We use the *insertion of nearest neighbor* procedure to build the tabu list by calculating the length of the giant tour created. The set of nodes that must be visited is now represented by the length of a tour constructed by procedure Insert1. The tabu list is initialized only once and has a length of θ_1, i.e., we will store all θ_1 solutions generated at the first phase. The computational results show that this list helps us to avoid re-exploiting 27.43% of the giant tours on average.

4.3. Resulting algorithm

Algorithm 1 gives the pseudocode for the resulting metaheuristic. Note that because the split procedure can handle the length constraints (see [25] for details), our algorithm can solve the general m-CTP problem.

5. Computational experiments

In this section, we describe the m-CTP-p instances and the computational evaluation of the proposed algorithm. Our algorithm is coded in C/C++ and is run on the same CPU used in [16], i.e., a 2.4-GHz CPU with 4 GB of RAM. The running time of the branch-and-cut algorithm is limited to 2 h for each instance.

The parameters θ_1, θ_2, and θ_3 in the metaheuristic are chosen so that they depend only on the problem data, i.e., they are generated according to an automatic mechanism. We tested many combinations and found that the following combination gives the best performance for our algorithm: $\{\theta_1, \theta_2, \theta_3\}$=$\{5(|V| - |T|), |N|, |N|\}$ where N is the set of required vertices generated by the first phase.

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

Data: name of instance;

Node: number of nodes in search tree of branch-and-cut algorithm;

Time: running time in seconds;

m: number of vehicles in solution;

Nv: number of vertices of V visited by the route in the optimal solution;

Imp: number of implied-bound cuts;

Go: number of Gomory cuts;

Do1: number of constraints of type (13);

Algorithm 1 Pseudocode for metaheuristic

1: $f(S_{final}) \Leftarrow +\infty$;
2: $Tabulist \Leftarrow \emptyset$;
3: **for** $t = 1 \rightarrow \theta_1$ **do**
4:     $N \Leftarrow$ initialize random MIP generator;
5:     **if** $N \in Tabulist$ **then**
6:         **go to** line 4;
7:     **else**
8:         add N to $Tabulist$;
9:     **end if**
10:     $L1 \Leftarrow \text{Insert1}(N)$;
11:     $S1 \Leftarrow \text{Split}(L1)$;
12:     $S1 \Leftarrow \text{LS1}(S1)$;
13:     $S^* \Leftarrow S1$;
14:     $L2 \Leftarrow \text{Insert2}(N)$;
15:     $S2 \Leftarrow \text{Split}(L2)$;
16:     $S2 \Leftarrow \text{LS1}(S1)$;
17:     **if** $f(S2) < f(S1)$ **then**
18:         $S^* \Leftarrow S2$;
19:     **end if**
20:     **for** $i = 1 \rightarrow \theta_2$ **do**
21:         $\overline{f} \Leftarrow +\infty$;
22:         **for** $j = 1 \rightarrow \theta_3$ **do**
23:             $L \Leftarrow \text{Concat}(S^*)$;
24:             $L \Leftarrow \text{Mutate}(L)$;
25:             $S \Leftarrow \text{Split}(L)$;
26:             $S \Leftarrow \text{LS1}(S)$;
27:             $S \Leftarrow \text{LS2}(S)$;
28:             $S \Leftarrow \text{LS3}(S)$;
29:             **if** $f(S) < \overline{f}$ **then**
30:                 $\overline{f} \Leftarrow f(S)$;
31:                 $\overline{S} \Leftarrow S$;
32:             **end if**
33:         **end for**
34:         **if** $\overline{f} < f(S^*)$ **then**
35:             $S^* \Leftarrow \overline{S}$;
36:         **end if**
37:     **end for**
38:     $S \Leftarrow S^*$;
39:     $S^* \Leftarrow \text{LS4}(S^*)$;
40:     **if** $f(S^*) < f(S)$ **then**
41:         **go to** line 20;
42:     **end if**
43:     **if** $f(S^*) < f(S_{final})$ **then**
44:         $S_{final} \Leftarrow S^*$;
45:         $f(S_{final}) \Leftarrow f(S^*)$;
46:     **end if**
47: **end for**

Do2: number of constraints of type (14) and (15);
Cov: number of constraints of type (16);
Flow: number of constraints of type (17);
Cap: number of constraints of type (18);
$LB0$: value of lower bounds at the root of the search tree (before adding the cuts);
$LB1$: value of lower bounds at the root of the search tree (after adding the cuts);
LB: the best lower bound in the branch-and-cut tree;
LS1: number of times LS1 is called;
LS2: number of times LS2 is called;
LS3: number of times LS3 is called;
LS4: number of times LS4 is called.

5.1. Data instances

We use the same procedure used in [16] to generate the instances for m-CTP-p. The instances kroA100, kroB100, kroC100, and kroD100 of TSPLIB are first used to create a set of $nb_{total} = V + W = 100$ vertices. Tests are run for $n = \lceil 0.25 nb_{total} \rceil$ and $\lceil 0.5 nb_{total} \rceil$ and $|T| = 1$ and $\lceil 0.20n \rceil$, and W is defined by taking the remaining points. The c_{ij} are computed as the Euclidean distances between the points. The value of c is determined so that each vertex of $V \setminus T$ covers at least one vertex of W, and each vertex of W is covered by at least two vertices of $V \setminus T$ (see [10, 16] for further information). We also use instances kroA200 and kroB200 with $nb_{total} = 200$ vertices to generate larger instances for m-CTP-p.

The instances are labeled X-T-n-W-p, where X is the name of the TSPLIB instance. For example, A2-1-50-150-4 indicates an instance derived from kroA200 of TSPLIB with 1 required vertex ($|T| = 1$), 50 vertices that can be visited ($|V| = 50$), 150 vertices that must be covered ($|W| = 150$), and $p = 4$.

5.2. Comparison with method of [16]

For a fair comparison with the exact algorithm of [16], we do not use the upper bound provided by the metaheuristic, and the running time for each instance is, as in [16], limited to 3600 s.

Table 1 gives the results of this experiment. The results in bold are proved to be optimal. The results of [16] which in some cases are not in bold (for example, the instance A1-25-75-6) mean that the method of [16] gives a solution not proven optimal. Jozefowiez gave some non-optimal solutions although the running time did not reach to limit because he developed an algorithm based on the column generation, not a branch-and-price one and the maximum number of columns searched at each iteration was limited to 25.

As can be seen, our method clearly outperforms that of [16]. Our branch-and-cut algorithm can solve all 32 instances, whereas the algorithm of [16] is unable to solve 10 instances. Our method is also faster on almost all of the successfully solved instances.

	Our method				Jozefowiez		
Data	m	Node	Time	Result	m	Time	Result
A1-25-75-4	2	10	1.23	**8479**	2	8	**8479**
A1-25-75-5	2	154	4.20	**8479**	2	10	**8479**
A1-25-75-6	2	628	10.90	**8479**	2	9	8724
A1-25-75-8	1	1180	13.32	**7985**	1	9	**7985**
A1-50-50-4	3	280	16.88	**10271**	3	252	**10271**
A1-50-50-5	2	250	15.67	**9220**	2	1156	**9220**
A1-50-50-6	2	1922	47.41	**9130**	2	1515	**9130**
A1-50-50-8	2	13345	228.61	**9130**	2	3600	11375
B1-25-75-4	2	30	2.28	**7146**	2	4	**7146**
B1-25-75-5	2	170	5.14	**6901**	2	8	**7013**
B1-25-75-6	1	115	4.65	**6450**	1	10	**6450**
B1-25-75-8	1	655	12.89	**6450**	1	11	**6450**
B1-50-50-4	2	360	21.35	**10107**	2	2297	**10107**
B1-50-50-5	2	3655	86.03	**9723**	2	2038	**9723**
B1-50-50-6	2	10970	229.12	**9382**	2	3600	9529
B1-50-50-8	2	3026	92.07	**8348**	1	3600	8701
C1-25-75-4	1	22	2.61	**6161**	1	3	**6161**
C1-25-75-5	1	149	5.24	**6161**	1	2	**6161**
C1-25-75-6	1	496	9.23	**6161**	1	3	**6161**
C1-25-75-8	1	1050	13.87	**6161**	1	2	**6161**
C1-50-50-4	3	504	24.69	**11372**	3	174	12156
C1-50-50-5	2	270	12.76	**9900**	2	1258	**9900**
C1-50-50-6	2	2915	65.35	**9895**	2	2169	10894
C1-50-50-8	2	114	11.72	**8699**	2	3450	8699
D1-25-75-4	2	12	1.33	**7671**	2	3	**7671**
D1-25-75-5	2	288	5.50	**7465**	2	15	7759
D1-25-75-6	1	147	4.87	**6651**	1	13	**6651**
D1-25-75-8	1	1713	24.10	**6651**	1	10	**6651**
D1-50-50-4	3	126	14.00	**11606**	3	239	**11606**
D1-50-50-5	2	882	38.22	**10770**	2	1562	**10770**
D1-50-50-6	2	7659	175.25	**10525**			
D1-50-50-8	2	5698	132.04	**9361**	3	3600	11703

Table 1: Comparison with method of [16]

5.3. Results for branch-and-cut algorithm

This subsection presents the results of the branch-and-cut algorithm in the case where the initial upper bounds provided by our metaheuristic are integrated as the bounds on the objective function. Let UB be the value of the final solution found by branch-and-cut algorithm or the value of the solution of the metaheuristic (if the branch-and-cut algorithm fails to find a solution), Gap_{BnC} in Tables 2 and 3 is computed as:

$$Gap_{BnC} = \frac{100.(UB - LB)}{UB} \tag{26}$$

Table 2 shows that our exact method can solve all but one of the instances with 100 vertices. Compared with the version without initial upper bounds, the number of nodes in the search tree is now lower in 25 of the 32 instances of [16]. For the 32 instances with 200 vertices (see Table 3), our algorithm can solve 20 instances successfully; most of them have $n = 50$. The problem difficulty increases with n and T but is fairly insensitive to $|W|$. This is similar to problem 1-CTP in [10]. Moreover, the greater the value of p, the harder the problem. Instances with $p = 4$ or 5 are usually solved more readily than instances with higher p values.

Tables 4 and 5 present the number of constraints generated in the branch-and-cut algorithm and the lower bounds at the root node of the search tree. In these tables, Gap_{LB} shows the deviation between the lower bounds LB0 and LB1 and is computed as:

$$Gap_{LB} = \frac{100.(LB1 - LB0)}{LB0} \tag{27}$$

Tables 4 and 5 clearly show the performance of valid inequalities in improving the linear relaxation of m-CTP-p, specially on the instances with $|T| = 1$. Among the cuts added, the flow constraints (17) are the most frequent; capacity constraints (18) are generated only when the value of $|T|$ is important.

5.4. Results for metaheuristic

Tables 6 and 7 report our results for the metaheuristic. The numbers in bold indicate the optimal solutions found by the branch-and-cut algorithm. The numbers marked with an asterix correspond to solutions that can not be improved by the branch-and-cut algorithm. Let UB be the value of the solution of the metaheuristic, Gap_{UB} in these tables reports the percentage deviation of the metaheuristic and is computed as:

$$Gap_{UB} = \frac{100.(UB - LB)}{UB} \tag{28}$$

where LB is the best lower bound in the branch-and-cut tree.

Our results confirm the quality of the metaheuristic. For the 82 instances for which the branch-and-cut algorithm can find an optimal solution, our metaheuristic can find an optimal solution in 79 cases. The optimality gaps for the three unsuccessful instances are small: 0.11%, 0.16%, and 1.45%. For the instances whose optimal solution is unknown, the branch-and-cut algorithm can not improve on the metaheuristic solution and we believe that the large gap G_{UB} in some cases are due to the poor quality of the lower bounds LB. Moreover, the running time is acceptable: it has a maximum of 126.00 s on the largest instance, kroB200-20-100-100-5.

6. Conclusion

In this paper, we have formulated and solved a particular case of the m-CTP where the length constraint is relaxed. We have presented an integer linear programming formulation that is solved using a branch-and-cut algorithm and developed a metaheuristic based on the ELS principle. We have reported computational results for a set of instances with up to 200 vertices where the tour contains up to 100 vertices. Our results clearly show the performance of our approach. Our branch-and-cut algorithm outperforms the algorithm of [16], and the solution provided by the metaheuristic is within 1.45% of optimality for the considered test instances.

The next step of our research will be to improve the branch-and-cut algorithm by adding more efficient cuts, so that we can evaluate our metaheuristic on larger instances. One line of investigation will be to study how the implied-bound cuts of CPLEX strengthen the linear relaxation of the model.

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] E. Balas and Shu Ming Ng. On the set covering polytope: I. All the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43(1-3):57–69, 1986.

[3] R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for 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.

[4] 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.

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

[6] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[7] J. Current. *Multiobjective design of transportation networks*. PhD dissertation, Johns Hopkins University, 1981.

[8] K-F. Doerner and R-F. Hartl. Health care logistics, emergency preparedness, and disater relief: New challenges for routing problems with a focus on the austrian situation. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 527–550. Springer, 2008.

[9] 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.

[10] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45:568–576, 1997.

[11] B. Gillett and L. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349, 1974.

[12] B. Golden, S. Raghavan, and E. Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

[13] M.-H. Ha, N. Bostel, A. Langevin, and L.-M. Rousseau. An exact algorithm for the close enough traveling salesman problem with arc covering

constraints. In *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems*, Vilamoura, Algarve, Portugal, February 2012.

[14] M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations Research*, 27:29–42, 2000.

[15] M-J. Hodgson, G. Laporte, and F. Semet. A covering tour model for planning mobile health care facilities in suhum district, gana. *Journal of regional science*, 38:621–638, 1998.

[16] N. Jozefowiez. A column generation approach for the multi-vehicle covering tour problem. In *Proceedings of Recherche Opérationnelle et Aide à la Décision Française (ROADEF 2011)*, Saint-Etienne, France, March 2011.

[17] 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.

[18] M. Labbé and G. Laporte. Maximizing user convenience and postal service efficiency in post box location. *Belgian journal of operations research, statistics and computer science*, 26:21–35, 1986.

[19] 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.

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

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

[22] 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.

[23] 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.

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

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

[26] M. Sánchez-García, M. I. Sobrón, and B. Vitoriano. On the set covering polytope: Facets with coefficients in {0, 1, 2, 3}. *Annals of Operations Research*, 81:343–356, 1998.

[27] J-C. Simms. Fixed and mobile facilities in dairy practice. *Veterinary clinics of North America - Food animal practice*, 5:591–601, 1989.

[28] W. Swaddiwudhipong, C. Chaovakiratipong, P. Nguntra, P. Lerdlukanavonge, and S. Koonchote. Effect of a mobile unit on changes in knowledge and use of cervical cancel screening among rural thai women. *International journal of epidemiology*, 24:493–498, 1995.

Data	m	Nv	Node	Time	Gap_{BnC}	Result
A1-1-25-75-4	2	8	7	1.13	0	8479
A1-1-25-75-5	2	8	111	3.27	0	8479
A1-1-25-75-6	2	8	441	6.87	0	8479
A1-1-25-75-8	1	8	1827	20.10	0	7985
A1-5-25-75-4	2	8	489	9.49	0	10827
A1-5-25-75-5	2	8	0	0.11	0	8659
A1-5-25-75-6	2	8	6	0.63	0	8659
A1-5-25-75-8	1	8	169	4.20	0	8265
A1-1-50-50-4	3	11	211	9.91	0	10271
A1-1-50-50-5	2	11	249	12.36	0	9220
A1-1-50-50-6	2	11	1223	24.79	0	9130
A1-1-50-50-8	2	11	12370	203.93	0	9130
A1-10-50-50-4	5	19	312215	4828.55	0	17953
A1-10-50-50-5	4	19	10554	173.61	0	15440
A1-10-50-50-6	3	19	123466	1586.21	0	14064
A1-10-50-50-8			357749	7200.12	5.61	
B1-1-25-75-4	2	7	20	1.81	0	7146
B1-1-25-75-5	2	7	87	3.23	0	6901
B1-1-25-75-6	1	7	102	4.33	0	6450
B1-1-25-75-8	1	7	593	10.88	0	6450
B1-5-25-75-4	2	9	0	0.22	0	9465
B1-5-25-75-5	2	9	178	5.74	0	9460
B1-5-25-75-6	2	9	1177	18.07	0	9148
B1-5-25-75-8	1	9	465	8.94	0	8306
B1-1-50-50-4	2	9	306	16.63	0	10107
B1-1-50-50-5	2	9	3642	84.08	0	9723
B1-1-50-50-6	2	10	8941	162.24	0	9382
B1-1-50-50-8	2	10	3862	76.06	0	8348
B1-10-50-50-4	4	17	8560	127.64	0	15209
B1-10-50-50-5	3	16	8504	149.24	0	13535
B1-10-50-50-6	3	16	6115	104.70	0	12067
B1-10-50-50-8	2	17	2000	32.27	0	10344
C1-1-25-75-4	1	5	21	2.82	0	6161
C1-1-25-75-5	1	5	177	5.81	0	6161
C1-1-25-75-6	1	5	289	7.73	0	6161
C1-1-25-75-8	1	5	577	9.42	0	6161
C1-5-25-75-4	2	9	6	0.39	0	9898
C1-5-25-75-5	2	9	112	2.98	0	9707
C1-5-25-75-6	2	9	170	4.17	0	9321
C1-5-25-75-8	2	9	1	0.35	0	7474
C1-1-50-50-4	3	11	258	8.12	0	11372
C1-1-50-50-5	2	10	354	13.28	0	9900
C1-1-50-50-6	2	11	2671	56.91	0	9895
C1-1-50-50-8	2	10	109	8.47	0	8699
C1-10-50-50-4	4	17	9647	164.37	0	18212
C1-10-50-50-5	4	17	8592	126.79	0	16362
C1-10-50-50-6	3	17	15289	240.39	0	14749
C1-10-50-50-8	2	17	303	5.62	0	12394
D1-1-25-75-4	2	7	13	1.04	0	7671
D1-1-25-75-5	2	7	256	5.38	0	7465
D1-1-25-75-6	1	7	89	3.80	0	6651
D1-1-25-75-8	1	7	1037	12.85	0	6651
D1-5-25-75-4	2	9	78	1.65	0	11820
D1-5-25-75-5	2	9	1626	16.72	0	10982
D1-5-25-75-6	2	10	150	3.40	0	9669
D1-5-25-75-8	1	9	129	1.25	0	8200
D1-1-50-50-4	3	10	95	9.34	0	11606
D1-1-50-50-5	2	11	938	29.32	0	10770
D1-1-50-50-6	2	11	12461	281.28	0	10525
D1-1-50-50-8	2	10	5222	110.62	0	9361
D1-10-50-50-4	5	19	393	10.93	0	20982
D1-10-50-50-5	4	18	27584	393.45	0	18576
D1-10-50-50-6	3	17	6574	116.08	0	16330
D1-10-50-50-8	3	17	15423	248.39	0	14204

Table 2: Computational results of branch-and-cut algorithm on instances with $nb_{total} = 100$

Data	m	Nv	Node	Time	Gap_{BnC}	Result
A2-1-50-150-4	2	9	150	82.21	0	11550
A2-1-50-150-5	2	10	1476	340.58	0	10407
A2-1-50-150-6	2	11	6498	1075.80	0	10068
A2-1-50-150-8	1	9	359	153.40	0	8896
A2-10-50-150-4	4	16	7108	1256.98	0	17083
A2-10-50-150-5	3	16	2374	494.66	0	14977
A2-10-50-150-6	3	16	4575	978.92	0	13894
A2-10-50-150-8	2	16	1164	280.21	0	11942
A2-1-100-100-4	3	10	25896	4593.91	0	11885
A2-1-100-100-5	2	11	6079	1440.13	0	10234
A2-1-100-100-6			23889	7200.13	5.85	
A2-1-100-100-8			24359	7200.12	12.88	
A2-20-100-100-4			43723	7200.08	1.97	
A2-20-100-100-5			30722	7200.13	4.38	
A2-20-100-100-6			31371	7200.14	4.43	
A2-20-100-100-8			27579	7200.08	8.76	
B2-1-50-150-4	3	10	253	166.00	0	11175
B2-1-50-150-5	2	10	4429	1114.67	0	10502
B2-1-50-150-6	2	10	6527	1273.97	0	9799
B2-1-50-150-8	2	10	2636	629.77	0	8846
B2-10-50-150-4	5	17	34309	5972.52	0	16667
B2-10-50-150-5	4	17	424	124.21	0	14188
B2-10-50-150-6	3	17	3893	773.56	0	12954
B2-10-50-150-8	2	17	3151	732.65	0	11495
B2-1-100-100-4	4	17	48551	6614.98	0	18370
B2-1-100-100-5	4	17	8181	1471.99	0	15876
B2-1-100-100-6			26450	7200.08	4.65	
B2-1-100-100-8			27427	7200.09	6.60	
B2-20-100-100-4			44254	7200.14	1.00	
B2-20-100-100-5			34898	7200.11	4.24	
B2-20-100-100-6			3536	7200.16	4.99	
B2-20-100-100-8			38336	7200.10	8.97	

Table 3: Computational results of branch-and-cut algorithm on instances with $nb_{total} = 200$

Data	Imp	Go	Do1	Do2	Cov	Flow	Cap	$LB0$	$LB1$	Gap_{LB}
A1-1-25-75-4	24	9	95	3	0	105	0	6018.33	8217.95	36.55
A1-1-25-75-5	32	11	135	66	0	157	0	5473.29	7521.08	37.41
A1-1-25-75-6	37	23	165	127	0	201	0	5100.11	7411.28	45.32
A1-1-25-75-8	34	21	195	161	0	309	0	4075.52	6340.25	55.57
A1-5-25-75-4	36	25	135	90	0	500	0	8052.23	9466.44	17.56
A1-5-25-75-5	8	16	11	1	0	8	0	7544.67	8659.00	14.77
A1-5-25-75-6	8	11	39	22	0	48	0	7243.45	8576.57	18.40
A1-5-25-75-8	25	14	78	76	0	125	0	6107.76	7398.16	21.13
A1-1-50-50-4	80	10	318	88	1	489	0	7395.66	9345.99	26.37
A1-1-50-50-5	98	9	330	160	1	503	0	5990.35	7925.38	32.30
A1-1-50-50-6	101	2	366	374	3	562	0	5449.91	7186.33	31.86
A1-1-50-50-8	121	8	491	617	1	756	0	4741.54	6325.92	33.41
A1-10-50-50-4	115	21	546	457	0	1062	0	14017.99	16223.01	15.73
A1-10-50-50-5	81	31	336	291	0	749	0	11943.33	14317.22	19.88
A1-10-50-50-6	81	40	445	226	2	913	0	10367.16	12769.89	23.18
A1-10-50-50-8	82	22	679	548	2	11382	6	9252.83	11280.34	21.91
B1-1-25-75-4	30	6	106	12	3	145	0	5070.24	6563.78	29.46
B1-1-25-75-5	30	6	124	38	3	173	0	4602.00	5821.71	26.50
B1-1-25-75-6	45	7	138	53	5	183	0	3835.19	5009.56	30.62
B1-1-25-75-8	36	9	201	90	9	296	0	3512.53	4464.54	27.10
B1-5-25-75-4	28	3	26	0	6	52	0	8198.25	9465.00	15.45
B1-5-25-75-5	23	12	94	43	6	180	0	7418.07	8713.15	17.46
B1-5-25-75-6	32	17	153	77	13	297	0	6877.75	8291.14	20.55
B1-5-25-75-8	27	11	95	51	8	220	0	5757.54	7193.69	24.94
B1-1-50-50-4	146	8	369	47	16	705	0	6546.09	8706.10	33.00
B1-1-50-50-5	166	12	501	249	20	977	0	5640.16	7750.26	37.41
B1-1-50-50-6	151	18	580	374	11	985	0	5038.11	7193.29	42.78
B1-1-50-50-8	161	6	532	395	27	779	0	3854.96	5744.45	49.01
B1-10-50-50-4	53	30	264	162	21	778	0	11940.07	14106.86	18.15
B1-10-50-50-5	52	27	314	186	23	802	0	10037.92	12125.23	20.79
B1-10-50-50-6	66	17	298	99	15	699	0	9106.06	10815.94	18.78
B1-10-50-50-8	56	25	218	62	18	516	3	7692.75	9342.99	21.45
C1-1-25-75-4	40	8	106	2	0	131	0	3824.38	5265.87	37.69
C1-1-25-75-5	44	9	133	50	2	177	0	3554.07	4946.48	39.18
C1-1-25-75-6	45	14	137	46	11	185	0	3364.92	4678.16	39.03
C1-1-25-75-8	41	12	157	120	4	588	0	3090.08	4279.73	38.50
C1-5-25-75-4	16	4	36	1	5	49	0	9010.92	9795.45	8.71
C1-5-25-75-5	23	10	87	29	21	125	0	8441.60	9205.33	9.05
C1-5-25-75-6	15	24	107	31	24	151	0	8077.32	8688.03	7.56
C1-5-25-75-8	16	3	43	0	0	63	0	6629.66	7474.00	12.74
C1-1-50-50-4	56	7	231	60	19	384	0	8435.52	10462.34	24.03
C1-1-50-50-5	74	10	315	120	9	360	0	6826.63	8896.96	30.33
C1-1-50-50-6	88	15	375	287	7	579	0	6242.41	8210.56	31.53
C1-1-50-50-8	56	14	234	81	6	294	0	5546.26	7479.82	34.86
C1-10-50-50-4	81	32	337	155	24	720	0	14892.18	17216.99	15.61
C1-10-50-50-5	65	24	283	237	18	590	0	13414.50	15583.38	16.17
C1-10-50-50-6	71	44	331	250	12	655	0	11648.63	13932.40	19.61
C1-10-50-50-8	31	11	102	12	0	209	0	9701.08	12000.56	23.70
D1-1-25-75-4	12	8	77	8	0	98	0	5665.90	7471.37	31.87
D1-1-25-75-5	24	7	109	42	0	197	0	5125.49	6630.58	29.36
D1-1-25-75-6	22	9	100	8	0	170	0	4088.89	5764.07	40.97
D1-1-25-75-8	27	13	172	98	0	235	0	3709.17	5187.67	39.86
D1-5-25-75-4	19	9	87	5	18	160	0	9291.02	11241.78	20.71
D1-5-25-75-5	51	18	171	43	18	298	0	8192.40	9870.06	20.48
D1-5-25-75-6	21	18	89	11	72	178	0	7521.79	8976.05	19.33
D1-5-25-75-8	16	9	66	1	18	119	0	6180.13	7516.38	21.62
D1-1-50-50-4	102	4	358	12	37	672	0	8180.47	10704.44	30.85
D1-1-50-50-5	124	7	430	129	38	747	0	6801.93	9139.56	34.37
D1-1-50-50-6	173	5	648	383	27	1048	0	5992.69	8264.90	37.92
D1-1-50-50-8	154	10	511	365	35	830	0	5013.83	6954.79	38.71
D1-10-50-50-4	50	12	250	45	9	640	5	17996.72	20346.85	13.06
D1-10-50-50-5	55	26	373	139	21	935	15	15024.75	17130.57	14.02
D1-10-50-50-6	59	23	309	136	11	822	0	12999.15	15073.76	20.81
D1-10-50-50-8	77	16	313	188	14	874	6	10775.99	12613.53	17.05

Table 4: Details of branch-and-cut algorithm on instances with $nb_{total} = 100$

Data	Imp	Go	Do1	Do2	Cov	Flow	Cap	$LB0$	$LB1$	Gap_{LB}
A2-1-50-150-4	86	8	235	14	31	407	0	7453.86	10437.83	40.03
A2-1-50-150-5	156	11	337	58	44	575	0	6512.04	9168.45	40.79
A2-1-50-150-6	179	18	454	122	56	712	0	5931.43	8527.62	43.77
A2-1-50-150-8	119	13	285	79	22	403	0	4510.18	7169.25	58.96
A2-10-50-150-4	189	25	283	76	46	677	0	14155.92	16049.60	13.38
A2-10-50-150-5	195	15	240	12	80	540	0	11994.84	13904.95	15.92
A2-10-50-150-6	138	31	274	145	68	588	0	11052.65	12959.76	17.25
A2-10-50-150-8	121	17	206	91	56	398	0	9224.62	11207.41	21.49
A2-1-100-100-4	742	2	1368	462	332	2301	0	7748.61	9675.91	24.87
A2-1-100-100-5	557	3	1121	410	259	1972	0	6355.55	8212.05	29.21
A2-1-100-100-6	619	3	1365	843	332	2578	0	5459.96	7162.93	31.19
A2-1-100-100-8	605	4	1491	1061	280	2657	0	4335.14	5883.03	35.71
A2-20-100-100-4	328	19	720	390	255	2232	150	21689.48	25042.62	15.46
A2-20-100-100-5	267	14	883	523	238	2486	75	18207.18	21331.41	17.16
A2-20-100-100-6	285	12	825	667	224	2245	40	15921.97	18775.14	17.92
A2-20-100-100-8	267	13	1028	767	238	2571	6	13150.13	15552.06	18.27
B2-1-50-150-4	137	6	373	39	95	624	0	6922.26	9572.96	38.29
B2-1-50-150-5	198	8	564	159	213	925	0	5918.43	8245.40	39.32
B2-1-50-150-6	184	24	514	193	94	884	0	5323.13	7604.10	42.85
B2-1-50-150-8	195	18	492	225	164	2643	0	4164.34	6213.36	49.20
B2-10-50-150-4	129	22	381	54	120	938	5	12762.43	15125.63	18.52
B2-10-50-150-5	56	26	158	19	78	339	0	11340.13	13574.24	19.70
B2-10-50-150-6	74	19	245	41	101	526	0	9971.28	11947.58	19.82
B2-10-50-150-8	80	22	245	52	132	503	0	8410.43	10078.25	19.83
B2-1-100-100-4	281	8	1246	847	30	2364	0	13110.87	16748.90	27.75
B2-1-100-100-5	248	8	856	750	15	1685	0	10989.93	14214.52	29.34
B2-1-100-100-6	296	7	1124	937	22	2542	0	9411.66	12394.72	31.70
B2-1-100-100-8	249	7	1207	1101	27	2311	0	7523.01	10366.04	37.79
B2-20-100-100-4	161	28	756	431	58	2241	31	29049.93	32913.34	13.30
B2-20-100-100-5	135	22	927	537	51	2620	96	23939.47	27494.07	14.85
B2-20-100-100-6	144	11	999	569	67	2711	82	20706.77	23954.74	15.69
B2-20-100-100-8	162	18	1090	647	52	3307	41	16794.95	19488.63	16.04

Table 5: Details of branch-and-cut algorithm on instances with $nb_{total} = 200$

Data	LS1	LS2	LS3	LS4	m	Time	Result	Gap_{UB}
A1-1-25-75-4	1675	15188	3010	332	2	0.16	**8479**	0
A1-1-25-75-5	410	18728	612	298	2	0.17	**8479**	0
A1-1-25-75-6	26	119554	1694	314	2	0.16	**8724**	0
A1-1-25-75-8	0	20502	374	319	1	0.16	**7985**	0
A1-5-25-75-4	396	2508	259	24	2	0.13	**10827**	0
A1-5-25-75-5	87	2504	70	20	2	0.14	**8659**	0
A1-5-25-75-6	4	2671	269	20	2	0.16	**8659**	0
A1-5-25-75-8	0	2628	0	19	1	0.14	**8265**	0
A1-1-50-50-4	4097	115543	18088	1161	3	0.80	**10271**	0
A1-1-50-50-5	10343	113686	25981	1141	2	0.78	**9220**	0
A1-1-50-50-6	8789	132666	768	1167	2	0.81	**9130**	0
A1-1-50-50-8	851	124757	4022	1168	1	0.81	**9130**	0
A1-10-50-50-4	19572	352699	91340	790	5	3.26	17973	0.11
A1-10-50-50-5	15487	396159	120965	879	4	3.81	**15440**	0
A1-10-50-50-6	13842	330348	248498	884	3	3.85	**14064**	0
A1-10-50-50-8	2851	454544	15607	816	3	3.92	13369*	5.61
B1-1-25-75-4	612	13666	1592	283	2	0.22	**7146**	0
B1-1-25-75-5	181	16526	2058	313	2	0.18	**6901**	0
B1-1-25-75-6	0	16822	277	311	1	0.23	**6450**	0
B1-1-25-75-8	0	16033	486	315	1	0.20	**6450**	0
B1-5-25-75-4	371	7900	5533	88	2	0.17	**9465**	0
B1-5-25-75-5	560	11756	1079	97	2	0.16	**9460**	0
B1-5-25-75-6	196	12082	822	81	2	0.17	**9148**	0
B1-5-25-75-8	0	14094	221	113	1	0.17	**8306**	0
B1-1-50-50-4	4722	48864	41466	999	2	0.62	**10107**	0
B1-1-50-50-5	4338	80211	5719	1065	2	0.64	**9723**	0
B1-1-50-50-6	1602	69445	4093	838	2	0.58	**9382**	0
B1-1-50-50-8	0	67874	1784	884	2	0.58	**8348**	0
B1-10-50-50-4	20284	203798	220064	763	4	2.53	**15209**	0
B1-10-50-50-5	2583	143178	51826	610	3	2.08	**13535**	0
B1-10-50-50-6	11045	237408	60456	604	3	1.97	**12067**	0
B1-10-50-50-8	10411	218846	119601	603	2	1.99	**10344**	0
C1-1-25-75-4	0	1493	40	75	1	0.16	**6161**	0
C1-1-25-75-5	0	2729	19	119	1	0.16	**6161**	0
C1-1-25-75-6	0	2729	19	119	1	0.15	**6161**	0
C1-1-25-75-8	0	2729	19	119	1	0.17	**6161**	0
C1-5-25-75-4	265	6570	6330	91	2	0.16	**9898**	0
C1-5-25-75-5	560	19909	2131	153	2	0.18	**9707**	0
C1-5-25-75-6	237	18507	1080	143	2	0.19	**9321**	0
C1-5-25-75-8	0	612694	47	122	1	0.19	**7474**	0
C1-1-50-50-4	729	76365	27063	992	3	0.64	**11372**	0
C1-1-50-50-5	2677	84615	18207	1099	2	0.67	**9900**	0
C1-1-50-50-6	1765	96358	1637	1039	2	0.67	**9895**	0
C1-1-50-50-8	180	91985	4164	1007	2	0.65	**8699**	0
C1-10-50-50-4	9962	174488	193441	651	4	2.23	**18212**	0
C1-10-50-50-5	2841	250973	85292	610	4	2.14	**16362**	0
C1-10-50-50-6	8750	272387	9479	698	3	2.00	**14749**	0
C1-10-50-50-8	5417	224571	137632	635	2	2.07	12414	0.16
D1-1-25-75-4	456	9332	1009	197	2	0.16	**7671**	0
D1-1-25-75-5	52	10340	738	178	2	0.16	**7465**	0
D1-1-25-75-6	0	11099	608	187	1	0.15	**6651**	0
D1-1-25-75-8	0	11074	585	212	1	0.16	**6651**	0
D1-5-25-75-4	87	8366	7010	77	2	0.18	**11820**	0
D1-5-25-75-5	460	13077	1443	74	2	0.17	**10982**	0
D1-5-25-75-6	363	14337	1076	85	2	0.17	**9669**	0
D1-5-25-75-8	10	16711	657	77	1	0.17	**8200**	0
D1-1-50-50-4	5405	122525	59874	1236	3	0.93	**11606**	0
D1-1-50-50-5	3834	120039	63496	1134	2	0.85	**10770**	0
D1-1-50-50-6	4361	135090	8384	1070	2	0.82	10680	1.45
D1-1-50-50-8	1563	149689	9205	1170	2	0.93	**9361**	0
D1-10-50-50-4	11357	276458	275638	731	5	3.82	**20982**	0
D1-10-50-50-5	8226	319468	156224	579	4	3.38	**18576**	0
D1-10-50-50-6	4823	276232	185681	450	3	2.91	**16330**	0
D1-10-50-50-8	5627	366309	157389	580	3	3.54	**14204**	0

Table 6: Computational results of metaheuristic on instances with $nb_{total} = 100$

Data	LS1	LS2	LS3	LS4	m	Time	Result	Gap_{UB}
A2-1-50-150-4	874	70356	34886	1024	2	0.89	**11550**	0
A2-1-50-150-5	2922	86127	15372	1014	2	0.87	**10407**	0
A2-1-50-150-6	1486	92947	1102	884	2	0.89	**10068**	0
A2-1-50-150-8	181	90790	5226	1069	1	0.94	**8896**	0
A2-10-50-150-4	9563	180071	126534	513	4	2.03	**17083**	0
A2-10-50-150-5	4457	167133	130439	462	3	1.50	**14977**	0
A2-10-50-150-6	6740	241410	26618	510	3	1.95	**13894**	0
A2-10-50-150-8	6093	234843	71875	566	2	2.07	**11942**	0
A2-1-100-100-4	6558	230334	53549	2791	3	2.89	**11885**	0
A2-1-100-100-5	7759	205319	110844	2839	2	2.82	**10234**	0
A2-1-100-100-6	6289	232113	4360	2533	2	2.92	10020*	5.85
A2-1-100-100-8	1556	253097	10516	2732	2	2.92	9093*	12.88
A2-20-100-100-4	57528	1328048	1656892	1613	7	43.91	26594*	1.97
A2-20-100-100-5	47550	1885843	746084	1417	6	37.29	23419*	4.38
A2-20-100-100-6	51174	2087397	712975	1583	5	39.50	20966*	4.43
A2-20-100-100-8	23543	2314063	295578	1748	4	42.42	18418*	8.76
B2-1-50-150-4	1593	67434	38854	776	3	0.91	**11175**	0
B2-1-50-150-5	5021	88174	15235	833	2	0.90	**10502**	0
B2-1-50-150-6	3080	91896	4423	877	2	0.91	**9799**	0
B2-1-50-150-8	170	80900	3311	766	2	0.87	**8846**	0
B2-10-50-150-4	14351	256685	136664	722	2	2.87	**16667**	0
B2-10-50-150-5	5410	282663	81587	773	2	2.78	**14188**	0
B2-10-50-150-6	9778	256081	35977	591	2	2.53	**12954**	0
B2-10-50-150-8	4927	269622	72082	595	1	2.53	**11495**	0
B2-1-100-100-4	51333	1045925	774566	4908	2	15.03	**18370**	0
B2-1-100-100-5	53439	1274929	614557	5073	2	15.61	**15876**	0
B2-1-100-100-6	28740	1152819	678806	4861	2	14.83	14926*	4.65
B2-1-100-100-8	22866	1348932	475145	4930	1	15.68	13137*	6.60
B2-20-100-100-4	144950	2947716	2554329	2542	9	117.01	34073*	1.00
B2-20-100-100-5	123354	3297460	2981437	2609	7	126.00	29412*	4.24
B2-20-100-100-6	93415	3827705	1968470	2784	6	116.79	25960*	4.99
B2-20-100-100-8	80471	3780534	1248722	2311	5	114.01	22156*	8.97

Table 7: Computational results of metaheuristic on instances with $nb_{total} = 200$