Solving the Close-Enough Arc Routing Problem

Minh Hoàng Hà
LUNAM Université, IRCCyN & Ecole des Mines de Nantes, IRCCyN, 4 rue Alfred Kastler, 44307 Nantes Cedex 3, France

Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, Quebec H3C 3A7, Canada

Nathalie Bostel
LUNAM Université, IRCCyN & Université de Nantes, 58 rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France

André Langevin and Louis-Martin Rousseau
Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, Quebec H3C 3A7, Canada

The close-enough arc routing problem has an interesting real-life application to routing for meter reading. In this article, we propose a new mathematical formulation for this problem. We analyze our formulation and compare it with two formulations in the literature. We also develop branch-and-cut algorithms to solve the problem to optimality. We present computational results for instances based on three types of graphs: directed, undirected, and mixed. © 2013 Wiley Periodicals, Inc. NETWORKS, Vol. 63(1), 107–118 2014

Keywords: close-enough arc routing problem; close-enough traveling salesman problem; automated meter reading; radio frequency identification; branch-and-cut algorithm

1. INTRODUCTION

Consider a directed graph $G = (V, A)$, with vertex set $V = \{v_0, v_1, \ldots, v_{n-1}\}$, and arc set of size m, $A = \{(v_i, v_j) : v_i, v_j \in V\}$. Numbering the arcs allows A to be expressed as $\{a_1, a_2, \ldots, a_m\}$. Let c_a be the cost associated with each arc of A. The well-known directed rural postman problem (DRPP) is the problem of determining a minimum-cost closed route traversing each arc in $A_r \subset A$ (called required arcs) at least once. Several algorithms have been proposed to solve the DRPP (see [1–3]). The closed-enough arc routing problem (CEARP) is a generalization of the DRPP in which the subset of required arcs A_r is not defined. Instead, there is a set of customers $W = \{w_1, w_2, \ldots, w_l\}$ that must be covered. These customers can be located anywhere in the area covered by the network, not only in the network itself. As in the original version of this problem (see [4]), we consider the inclusion of a depot. By definition, vertex v_0 is this depot. The CEARP consists in finding a minimum-cost tour, which begins and ends at the depot, such that every customer of W is covered by the tour, that is, lies within a distance r of an arc of the tour.

The main application of this problem is to construct routes for meter reading (see [4, 5] for further information). In this application, to measure customers' consumption information from a distance, a reading device is installed in a vehicle to collect the data sent by metering devices. The reader does not have to visit each customer to collect the data but must enter the meter's read range. The vehicle must traverse the service area and pass close enough to each meter so that they can all be read. The effective radius r, also called the read range, is normally between 150 and 300 m but may be as high as 381 m (see [4]).

The CEARP is clearly NP-hard because it contains the DRPP as a special case. The existing literature on the CEARP is limited. The earliest work on the CEARP is [4]. The authors call the problem a close-enough traveling salesman problem (CETSP) over a street network and propose four heuristics to solve 18 real-life instances with an average of about 900 street segments and 9000 customers each. The read range r is tested with two values: about 100 and 150 m, respectively. Basically, the heuristics of [4] are implemented in a two-stage process. In stage one, the heuristic identifies a subset of arcs to be traversed, either with some simple greedy

Received February, 2012; accepted November, 2012
Correspondence to: L. M. Rousseau; e-mail: louis-martin.rousseau@polymtl.ca
DOI 10.1002/net.21525
Published online 1 October 2013 in Wiley Online Library (wileyonlinelibrary.com).

procedures or by solving an integer program. In stage two, the problem becomes the well-known DRPP and is solved by a sophisticated heuristic.

To the best of our knowledge, the only exact method for the CEARP is [5]. The authors propose an initial mixed integer programming (MIP) formulation for the problem in which there is a connectivity constraint, and they solve this optimally using a cutting-plane approach. The algorithm first solves the problem without the connectivity constraints. The violated connectivity constraints are then added to the model, and the process is repeated until the connectivity is satisfied. Computational results show that this algorithm can solve to optimality random instances of a realistic size, such as those introduced in [4].

The CEARP is equivalent to the generalized DRPP (GDRPP). In the GDRPP, there are several subsets of arcs (also called clusters) and the objective is to find a minimum-cost tour traversing at least one arc from each cluster. The clusters may be connected or disjoint. In the CEARP, the arcs covering a customer correspond to a subset in the GDRPP. The version of the GDRPP with no depot is described in [6], where both exact and heuristic methods are presented. The exact method is a branch-and-cut algorithm based on a mathematical formulation.

There is another problem that can be seen as a CETSP with vertex-covering constraints. It is the covering tour problem (CTP) [7] and is similar to the CEARP except that a closed tour has to be determined so that every vertex of W lies within a distance r of a *vertex* of the tour. In [7], an exact algorithm and a heuristic are presented. A heuristic based on the scatter-search method has been proposed in [8]. Heuristics for the multi-vehicle version (m-CTP) are presented in [9]. Heuristics have also been proposed for problems belonging to the family of CTPs, such as the CETSP in the plane; see [10, 11].

The aim of this article is to design exact algorithms for the CEARP. Our main contributions are: (1) we introduce a new formulation for the CEARP; (2) we compare, both analytically and empirically, this formulation with two others, the first introduced in [5] and the second presented in [6] for the GDRPP; (3) we improve the branch-and-cut algorithm of [6] and propose a new algorithm for the CEARP; and (4) we propose a MIP-based constructive algorithm to solve the CEARP in practice.

The remainder of the article is organized as follows. Section 2 presents in detail our branch-and-cut algorithms and a MIP-based constructive algorithm to solve the CEARP. The computational results are reported and analyzed in section 3. Finally, section 4 summarizes our conclusions.

2. BRANCH-AND-CUT ALGORITHMS FOR CEARP

2.1. Mathematical Formulations

Given a node subset, $S \subseteq V$, let $\delta^+(S)$ denote the set of outgoing arcs of S and $\delta^-(S)$ denote the set of incoming arcs of S. If $S = \{v_k\}$, we simply write $\delta^+(k)$ (or $\delta^-(k)$) instead of $\delta^+(\{v_k\})$ (or $\delta^-(\{v_k\})$). $A(S)$ is the set of arcs with both endpoints in S. Each arc a is associated with a cost c_a (distance or travel time). We define the binary coefficients λ_{lk} to be equal to 1 if and only if $w_l \in W$ can be covered by $a_k \in A$. Given $x \in N^{|A|}$ and $T \subset A$, $x(T)$ denotes $\sum_{e \in T} x_e$.

In [5], the following formulation, denoted by F1, was presented:

$$\text{F1:} \quad \text{Minimize} \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{subject to} \quad x\left(\delta^+(0)\right) \geq 1 \tag{2}$$

$$x\left(\delta^+(i)\right) - x\left(\delta^-(i)\right) = 0 \qquad \forall i \in V \tag{3}$$

$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{4}$$

$$Mx\left(\delta^+(S)\right) - x_a \geq 0 \qquad \forall S \subset V - \{v_0\} \text{ and } 2 \leq |S| \leq n-2,\ a \in A(S) \tag{5}$$

$$x_a \in Z^+ \qquad \forall a \in A \tag{6}$$

where M is a large number representing an upper bound on the number of times an arc is used. As in [6], M is set to $|A| + 1$.

The objective (1) minimizes total travel cost. Constraint (2) ensures that the depot belongs to the tour, and constraints (3) are the flow conservation constraints. Constraints (4) ensure that every customer of W is covered by the tour, and constraints (5) are the connectivity constraints. These constraints ensure the presence of at least one outgoing arc of any set S, for every possible subset S of V containing an arc belonging to the tour. Constraints (6) define the variables.

As, the CEARP is equivalent to the GDRPP, the formulation in [6] proposed for the GDRPP can be used for the CEARP. Because the formulation in [6] addresses the no-depot version of the problem, we modify it slightly. In this formulation, y_i is a binary variable that indicates the use of vertex i in the solution, and the integer variable x_a denotes the number of times that arc a is traversed. Let he_a be the head of arc a. The formulation for the CEARP is as follows:

$$\text{F2:} \quad \text{Minimize} \quad \sum_{a \in A} c_a x_a \tag{7}$$

$$\text{subject to} \quad y_0 = 1 \tag{8}$$

$$x\left(\delta^+(i)\right) - x\left(\delta^-(i)\right) = 0 \qquad \forall i \in V \tag{9}$$

$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{10}$$

$$x\left(\delta^+(S)\right) - y_i \geq 0 \qquad \forall S \subset V - \{v_0\}, \text{ and } 2 \leq |S| \leq n-2,\ i \in S \tag{11}$$

$$x_a - My_i \leq 0 \qquad \forall i \in V, a \in A \text{ with } he_a = i \tag{12}$$

$$x_a \text{ positive integer and } y_i \in \{0,1\} \quad \forall a \in A \text{ and } \forall i \in V \tag{13}$$

where M is a large number representing an upper bound on the number of times an arc is used. In this formulation, constraints (8)–(11) have the same meaning as in formulation F1, and constraints (12) are used to express the relation between the two types of variables. As in [6], M is set to $|A| + 1$.

We now describe a new formulation for the CEARP, which we call F3. Let y_a be a binary variable that represents the use of arc a with service. The integer variable x_a denotes the number of times that arc a is used without service. Then the CEARP can be stated as:

$$\text{F3:} \quad \text{Minimize} \quad \sum_{a \in A} c_a(x_a + y_a) \tag{14}$$

$$\text{subject to} \quad x\Big(\delta^+(0)\Big) + y\Big(\delta^+(0)\Big) \geq 1 \tag{15}$$

$$x\Big(\delta^+(i)\Big) + y\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) - y\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{16}$$

$$\sum_{a \in A} \lambda_{wa} y_a \geq 1 \qquad \forall w \in W \tag{17}$$

$$x\Big(\delta^+(S)\Big) + y\Big(\delta^+(S)\Big) - y_a \geq 0 \qquad \forall S \subset V - \{v_0\}, \text{ and } 2 \leq |S| \leq n-2,\ a \in A(S) \tag{18}$$

$$x_a \text{ positive integer and } y_a \in \{0,1\} \qquad \forall a \in A. \tag{19}$$

The meaning of each constraint in F3 is as in F1. Constraints (15)–(18) imply constraints (2)–(5), respectively.

It is easy to see that the three formulations F1, F2, and F3 are equivalent. Table 1 gives a comparison of the three formulations. F1 has the smallest number of variables and constraints (we do not consider the connectivity constraints here because of their exponential number). The most important disadvantage of F1 is that it uses a large number M in the connectivity constraint. Because we can not find an efficient way to closely estimate M, an exact procedure to separate this constraint is useless. Experiments show that the performance of the branch-and-cut algorithm based on this formulation is poor compared to that of the other two formulations. F3 has the largest number of variables, but it also has some strong advantages. First, it does not contain the Big-M constraints that are known to weaken the linear relaxation and to decrease the performance of MILP models. Second, the set covering polytope with binary variables has received more attention than that with integer variables. Hence, the identification of violated constraints of type (17) in F3 seems to be more favorable than that for (4) in F1 or (10) in F2. Formulation F2 has fewer variables but more constraints than F3. Moreover, it contains Big-M constraints.

2.2. *Valid Equalities and Inequalities*

In this subsection, we introduce several valid equalities and inequalities for F3. Let $A' \subset A$ be a set of arcs that can not cover any customer. The following dominance relation is valid for the CEARP:

TABLE 1. Comparison of the three formulations.

	F1	F2	F3
No. variables	$\|A\|$	$\|A\| + \|V\|$	$2.\|A\|$
No. constraints	$\|V\| + \|W\| + 1$	$\|V\| + \|A\| + \|W\| + 1$	$\|V\| + \|W\| + 1$
Big-M	Yes	Yes	No

$$y_a = 0 \qquad \forall a \in A'. \tag{20}$$

Lemma 1. *Without loss of generality, we can assume that in an optimal solution $x_a > 0$ implies $y_a = 1\ \forall\ a \notin A'$.*

Proof. Suppose that (x^*, y^*) is an optimal solution such that $x^*_{a'} > 0$ and $y^*_{a'} = 0$ for some arc $a' \notin A'$. Define $(x_a, y_a) = (x^*_a, y^*_a), \forall a \neq a'$ and $(x'_a, y'_a) = (x^*_{a'} - 1, 1)$. It is easy to prove that this new solution is also optimal because it satisfies the constraints (15), (16), (17), (18) and has the same objective value (14). ∎

We can derive from Lemma 1 the following dominance property:

$$My_a \geq x_a \qquad \forall a \in A \backslash A' \tag{21}$$

where M is a large number representing an upper bound on the number of times an arc a is used. Its value can be taken as $|A|$.

Dominance constraints can also be derived, based on covering considerations. Let $a_i, a_j \in A \backslash A'$. Arc a_i is said to dominate arc a_j if for all $w_l \in W$, $\lambda_{il} \geq \lambda_{jl}$. The following constraints proposed in [7] are valid for the CEARP:

$$y_i + y_j \leq 1 \qquad \forall a_i, a_j \in A \backslash A' \text{ and if } a_i \text{ dominates } a_j, \text{ or conversely.} \tag{22}$$

We also note that the constraints (21) and (22) can not be used at the same time because of contrary properties. The reason is that the former implies that if an arc that can cover a customer is traversed, then it is always used with service while the latter limits the use of arcs with service only when necessary.

All the valid inequalities of the set covering polytope $conv\{y : \sum b_a.y_a \geq 1, y_a \in \{0,1\}\}$ are valid for the CEARP. Balas and Ng [12] proposed the facets with coefficients in {0,1,2} and Sánchez-García et al. [13] introduced the more complex facets with coefficients in {0,1,2,3}. Here, we recall the first inequality that was used in [7]. Let S be a nonempty subset of W and define for each $a \in A$ the coefficient

$$\alpha^S_a = \begin{cases} 0 \ \texttt{if} \ \lambda_{al} = 0 \ \texttt{for all} \ w_l \in S, \\ 2 \ \texttt{if} \ \lambda_{al} = 1 \ \texttt{for all} \ w_l \in S, \\ 1 \ \texttt{otherwise}. \end{cases}$$

Then the following inequality is valid for the CEARP:

$$\sum_{a \in A} \alpha^S_a y_a \geq 2 \qquad \forall a \in A \backslash A' \text{ and } \forall S \subset W : |S| > 0. \tag{23}$$

The connectivity constraints (18) can be strengthened by adding information about the covering constraint. The following constraint is a stronger form of connectivity:

$$x\left(\delta^+(S)\right) + y\left(\delta^+(S)\right) \geq 1 \tag{24}$$

where $S \subset V$ and at least one customer of W is not covered by any arc in $A(V\backslash S)$.

2.3. Separation of Cuts

We now present separation procedures for each class of valid inequalities described in the previous subsection. To simplify the presentation, we give the details only for F3. The separation of cuts for F2 is similar. Let (x^*, y^*) be a fractional solution to be separated. Let G^* be the weighted graph induced by (x^*, y^*) such that the capacity of each arc $C_a = x_a^* + y_a^*$.

The detection of constraints of type (20), (21), and (22) is straightforward. Although these constraints help to reduce the number of vertices in the branch-and-bound tree, constraints (21) contain the Big-M that decreases the performance of the branch-and-cut algorithm. Since M is determined by the number of arcs, the larger the instance, the more important the negative impact of M. Our tests show that when M can not be closely estimated, a branch-and-cut algorithm without Big-M constraints is normally more efficient than one with Big-M. This is why we use the constraints (20) and (22) in the general case where M can not be estimated effectively. The constraints (21) are only used if we can find the way to bound M strictly. Whenever these dominance constraints are used, they are directly included in the initial model because they are almost certainly violated.

For constraints (23), we tested $|S| = 3$, as in [7], to reduce the computational effort. The process was still time-consuming because of the large number of customers in the CEARP instances. However, we observe that the zero-half cut of CPLEX can generate this type of constraint. Our tests show that using the CPLEX cut is more effective and faster than directly applying (23).

We will present an example to show how a zero-half cut can generate the constraints (23). Suppose there is a graph with three arcs a_1, a_2, and a_3 serving three customers w_1, w_2, and w_3. These customers can be served by the arcs (a_2, a_3), (a_1, a_3), and (a_1, a_2), respectively. The covering constraints (17) for three customers are as follows: $y_2 + y_3 \geq 1, y_1 + y_3 \geq 1$, and $y_1 + y_2 \geq 1$. By considering the sum of these three constraints, dividing the resulting constraint by 2, and rounding its right-hand side, we get the zero-half cut: $y_1 + y_2 + y_3 \geq 2$. This constraint has the same form as (23). Hence, the CPLEX zero-half cut can generate the constraints (23).

The connectivity constraints (18) can be separated by both heuristic and exact methods. In the heuristic method, we first compute the connected components $S_1, S_2, \ldots, S_q$ of the subgraph G^*. If the number of connected components is at least two (i.e., $q \geq 2$) then $x(\delta^+(S_i)) + y(\delta^+(S_i)) \geq y_a$, for each component i and each $a \in A(S_i)$, is a violated inequality. We can also separate these constraints exactly in polynomial time by applying the following algorithm:

- For each node $i \in V\backslash\{v_0\}$: Compute the maximum flow from i to depot v_0. Let $\delta^+(S_i)$, $S_i \subseteq V\backslash\{v_0\}$ be the min-cut and let f_i be the value of this flow.
- For each arc $a = (v_i, v_j) \in A$ and $x_a \neq 0$: If $f_i < x_a$, then a violated connectivity constraint has been found.

Let f_a be the maximum flow passing through arc $a = (v_i, v_j)$. To ensure connectivity from the depot to arc a, there must be at least a flow of x_a passing through a. Note that $f_i \geq f_a$. Then, if $f_i < x_a$, we have $f_a < x_a$. Hence, $f_i < x_a$ is a violated constraint. Conversely, if $f_i \geq x_a$, we can always send a flow of at least x_a through a and connectivity is ensured. If the Edmonds-Karp algorithm, which runs in $\mathcal{O}(|V||A|^2)$ time, is used to solve the maximum flow problems, the running time of this procedure is $\mathcal{O}(|V|^2|A|^2)$.

Finally, constraints (24) can be generated by a heuristic similar to that proposed for (18). Compute the connected components $S_1, S_2, \ldots, S_q$ of the subgraph $G(x^*, y^*)$ induced by the arcs $a \in A$ with $x^* + y^* \geq \epsilon$, where $0 \leq \epsilon < 1$ is a given parameter. If the number of connected components is at least two (i.e., $q \geq 2$), and the arcs with two endpoints in $V\backslash S$ can not cover all the customers and $x^*(\delta^+(S_i)) + y^*(\delta^+(S_i)) < 1$ then $x(\delta^+(S_i)) + y(\delta^+(S_i)) \geq 1$ for each component $i = 1, \ldots, q$ is a violated inequality.

2.4. Upper Bound for CEARP

In this subsection, we introduce a fast heuristic called UB1 that gives feasible solutions for the CEARP. This heuristic is used in the exact algorithm to provide an initial upper bound. It is based on the algorithm proposed for the DRPP [2] and is as follows:

STEP 1. Solve an integer program including the constraints (1), (3), (4), and (6) using CPLEX. The solution of this MIP is rapid even for the large instances in our tests.

STEP 2. Construct a directed graph $G_R = (V_R, A_R)$ induced by the solution from Step 1, adding the depot if it is not already present. If G_R is connected then stop; the solution found in Step 1 is also a feasible solution for the CEARP. Otherwise, go to Step 3.

STEP 3. Compute the connected components $C_1, C_2, \ldots, C_k$ of the graph G_R. Let K_i be the set of vertices corresponding to the connected component i. Build the undirected graph $\bar{G} = (N, E)$ with set of vertices $N = 1, \ldots, k$ and set of edges $E = \{(i,j), i,j \in N, i \neq j$. The corresponding edge costs are $c_{ij} = \min\{c(p,q) + c(q,p), p \in K_i, q \in K_j\}$ for each edge $(i,j) \in E$, where $c(p,q)$ is the length of the shortest path from vertex p in component i to vertex q in component j. Determine a minimum-cost spanning tree in $\bar{G}$. Let A_T be the set of arcs in the original graph corresponding to the edges in the tree. Add to $\bar{G}$ all the arcs of A_T.

TABLE 2. Comparison of lower bounds of the four algorithms.

Instance	lb_1	lb_2	lb_3	lb_4
ce500-1500-150-0.5-1	97,308.9	100,406.7	**102,417.6**	97,320.2
ce500-1500-150-0.5-2	91,374.4	**94,306.2**	93,393.5	90,309.9
ce500-1500-150-0.5-3	105,410.5	109,439.3	**109,441.2**	102,412.4
ce500-1500-150-0.5-4	89,307.1	**92,352.3**	92,323.0	86,379.5
ce500-1500-150-0.5-5	97,380.9	99,376.9	**100,412.6**	93,386.0
ce500-1500-150-1-1	124,465.7	126,487.0	**127,417.0**	122,482.1
ce500-1500-150-1-2	119,485.3	121,420.3	**122,407.2**	116,432.0
ce500-1500-150-1-3	129,484.2	**131,444.9**	131,422.7	126,413.8
ce500-1500-150-1-4	109,407.3	112,458.5	**113,458.1**	108,449.6
ce500-1500-150-1-5	112,412.1	**114,481.8**	113,400.8	106,413.5
ce500-1500-150-5-1	158,493.7	160,443.6	**160,472.9**	157,472.1
ce500-1500-150-5-2	158,452.9	159,478.5	**159,497.7**	157,400.3
ce500-1500-150-5-3	174,484.4	**176,492.5**	176,466.2	171,483.5
ce500-1500-150-5-4	149,410.1	150,454.0	**150,465.4**	147,412.2
ce500-1500-150-5-5	159,417.1	161,405.9	**161,436.8**	156,457.5
ce500-1500-150-10-1	172,404.7	173,457.7	**174,419.1**	170,485.1
ce500-1500-150-10-2	171,401.5	172,417.5	**172,491.4**	169,447.9
ce500-1500-150-10-3	183,425.9	184,483.5	**184,408.3**	180,471.4
ce500-1500-150-10-4	160,415.3	161,400.2	**161,479.4**	157,474.1
ce500-1500-150-10-5	166,463.3	166,489.9	**167,419.9**	163,420.5
ce500-1000-200-0.5-1	68,366.5	70,397.9	**73,387.9**	65,320.4
ce500-1000-200-0.5-2	75,373.2	79,338.8	**82,308.3**	74,342.7
ce500-1000-200-0.5-3	75,331.4	79,391.9	**86,363.8**	70,341.0
ce500-1000-200-0.5-4	64,385.7	72,382.7	**73,316.8**	68,379.2
ce500-1000-200-0.5-5	78,365.6	**83,302.2**	83,382.8	79,326.7
ce500-1000-200-1-1	75,301.9	79,380.2	**80,334.8**	71,399.5
ce500-1000-200-1-2	83,330.2	85,311.7	**88,301.9**	75,334.9
ce500-1000-200-1-3	90,327.1	94,397.9	**95,375.6**	90,369.7
ce500-1000-200-1-4	73,371.1	77,359.5	**79,349.0**	75,344.6
ce500-1000-200-1-5	87,319.4	**91,369.9**	90,390.5	88,367.4
ce500-1000-200-5-1	94,393.6	98,307.2	**98,347.0**	93,316.3
ce500-1000-200-5-2	105,408.9	108,494.2	**108,484.0**	104,436.4
ce500-1000-200-5-3	110,431.7	**113,453.4**	112,452.4	112,429.9
ce500-1000-200-5-4	98,342.1	99,383.7	**103,466.7**	97,315.6
ce500-1000-200-5-5	108,461.5	111,496.9	**111,432.5**	110,469.1
ce500-1000-200-10-1	108,402.9	**110,422.7**	109,462.8	107,473.1
ce500-1000-200-10-2	108,427.2	**112,465.9**	112,429.0	110,469.1
ce500-1000-200-10-3	123,441.6	125,444.3	**125,447.7**	124,449.3
ce500-1000-200-10-4	111,434.0	112,484.6	**115,422.7**	111,402.0
ce500-1000-200-10-5	124,483.3	126,452.4	**127,471.7**	124,476.1

2.5. Branch-and-Cut Algorithms

We first introduce a procedure that can reduce the number of customers. Given $w_l \in W$, let $Z(w_l)$ be the set of arcs that can cover w_l. For each pair of customers w_i and w_j, if $Z(w_i) \subseteq Z(w_j)$ then customer w_j can be ignored. This is because when w_i is served, w_j is covered at the same time. Note that the number of customers remaining is also the maximum number of arcs that must be activated for covering purposes. This procedure eliminates at least 50% of the customers in our tests.

We solve the CEARP exactly using a classic branch-and-cut algorithm. To simplify the description we describe the algorithm only for F3; the implementation is similar for F1 and F2. We solve a linear program containing the constraints (15), (16), (17), (20), (21) [or (22)], and constraints $0 \leq y_a \leq 1$. We then search for violated constraints of type (18) and (24), and the constraints detected 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 to generate two new subproblems. If all the variables are integer, we explore another subproblem.

Because the exact separation for (18) is quite time-consuming, after numerous tests, we decided to apply it only at the root node. More precisely, at every node, we first find the strongly connected components of the graph created by the current variables. For each component, we check which customers are covered. If these customers can not be covered by the arcs outside the component, a constraint of type (24) is found. Otherwise, a constraint of type (18) is detected and added to the current program. At the root node, if this procedure fails to find violated constraints, we carry out the exact separation method. In other words, the exact separation of constraint (18) is used at the root node only if the heuristic fails. At the other nodes, only heuristic separation is applied. For F1, as we can not find any way to closely estimate M in constraints (5), the exact separation is useless for these constraints. Therefore, only the heuristic method is used even at the root node.

Our branch-and-cut algorithm is built around CPLEX 11.2 with the Callable Library. All CPLEX cuts except the zero-half cut are turned off. The parameter CPX–PARAM–ZEROHALFCUTS is set to 2 to generate zero-half cuts aggressively. 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 [7] and branching on the variables x before y, but these do not outperform the CPLEX branching. Hence, we let CPLEX make the branching decisions.

In [6], a branch-and-cut algorithm based on F2 is developed to solve the GDRPP. The differences with our algorithm are that all the cuts of CPLEX 9.1 are turned on by default, the constraints (24) are not used, and the exact method to separate the connectivity constraints (11) is used at all nodes in the search tree. We also note that zero-half cuts were not implemented in CPLEX 9.1.

2.6. A MIP-Based Constructive Algorithm

In this subsection, we propose an algorithm called UB2 that gives good solutions for the CEARP. In practice, arcs are

TABLE 3. Number of successful instances.

Data	F0	F1	F2	F3	F4
ce500-1500-0.5	0	0	0	0	0
ce500-1500-1	1	0	2	2	0
ce500-1500-5	4	3	5	5	2
ce500-1500-10	5	5	5	5	4
ce500-1000-0.5	2	0	1	3	1
ce500-1000-1	3	2	4	4	2
ce500-1000-5	5	3	5	5	4
ce500-1000-10	5	5	5	5	5

usually traversed just a few times, and the number of traversals is much lower than the lowest provable value of M. This suggests that we can bound the large number M in the branch-and-cut algorithm by a small value to improve performance. To ensure that a solution exists, a Chinese postman problem (CPP) is first solved, and M is determined by the maximum number of times an arc is traversed in the CPP solution. We use only F2 and F3 to construct this algorithm. For F3, the dominance constraint (21) is used and added directly to the initial model.

3. COMPUTATIONAL EXPERIMENTS

In this section, we describe the CEARP instances and the computational evaluation of the proposed algorithms. Our algorithm is coded in C/C++ and is run on a 2.4-GHz CPU with 6 GB of RAM. The running time of the branch-and-cut algorithms is limited to 2 h for each instance.

3.1. Data Instances

We first use the CEARP instances of [5], which are random instances based on directed graphs. We now recall how to build these instances. Graphs that imitate real street networks are first generated randomly; this procedure is as follows:

- The coordinates of n vertices are randomly generated in a unit square. Then a heuristic is used to find the shortest tour passing through all the nodes exactly once. This tour is a Hamiltonian circuit, and it is used as a framework to construct the full graph. The resulting graph is therefore strongly connected.
- To imitate real networks, random arcs are added to the current tour to give a total of $m = nd$ arcs, where n is the number of vertices and d the ratio between the number of arcs and the number of nodes, in such a way that: (i) the arcs are not too long and (ii) there is no intersection between any two arcs.

In the tests in [5], graphs with $n \in \{300, 400, 500\}$ vertices and a ratio $d \in \{1.5, 2, 2.5, 3\}$ are used. For each combination of n and d, five different graphs are generated. The cost of an arc (v_i, v_j) is the Euclidean distance between v_i and v_j multiplied by five to obtain an average arc length close to that seen in practice (from about 0.2 to 0.4 km).

Once the graphs have been built, the CEARP instances are generated by randomly positioning $q = mt$ customer nodes in the square containing the graph, where m is the number of arcs and t the ratio between the number of customers and the number of arcs, $t \in \{0.5, 1, 5, 10\}$. Thus, for each graph, four CEARP instances are created. The effective radius r is set to 150 m. The distance between the arcs and the customer vertices are computed by the distance from the closest point of the arc to the customer. To ensure that a solution exists, we delete all the customers that can not be covered by any arc. We also examine the impact of increasing the radius parameter from 150 to 200 m. To do this, we use the graph created with $r = 150$ m but change the read range to 200 m.

From the instances of [5], we choose the largest ones with 500 nodes and 1500 arcs (r=150 m), and 500 nodes and 1000 arcs (r=200 m) to test our algorithms. The instances are labeled ce-n-k-r-t-i, where n is the number of nodes, k is the number of arcs, r is the read range, t is the ratio between the number of customers and the number of arcs, and i (=1,…,5) is the instance number. For example, ce-500-1500-150-10-5 indicates the fifth instance with 500 nodes, 1500 arcs, 150 m of read range, and t =10.

We also use mixed graphs from the literature to generate instances for the CEARP. We choose the mixed graphs introduced in [14] for which the coordinates of the vertices are published. These are large graphs with a structure similar to that of real street networks. To transform the mixed graphs to directed graphs, we model each undirected edge by two arcs with the same cost. From these graphs, we select two, MB537 and MB547, which, after being transformed to directed graphs, have fewer than 1500 arcs. MB357 has 500 nodes, 364 edges, and 476 arcs and MB547 has 500 nodes, 351 edges, and 681 arcs.

For the mixed graphs, the procedure to generate the customers is the same, except that the read range r is determined by the average length of all the arcs in the graph. For each graph and each value of t, we generate five CEARP instances.

We also test our algorithms on 30 instances defined on undirected graphs. We use two sets of graphs taken from [15] with 15 general routing problem instances each, generated from the Albaida and Madrigueras graphs proposed in [16] by defining each edge as being required with probability $p = \{0.3, 0.5, 0.7\}$. The Albaida graph includes 116 nodes and 174 edges, and the Madrigueras graph has 196 nodes and 316 edges. From these graphs, we generate CEARP instances as follows:

- The number of clients is defined by the number of required edges in the graph; each client is covered by a required edge.
- Each client is covered by e additional required edges where e is a random number taking values from 1 to 5, so that each client is covered by at least 2 and at most 6 edges.

To solve the undirected instances as directed instances, we transform each edge into two arcs with the same cost. As in other arc routing problems defined on undirected graphs (see [17], e.g.), it is easy to prove that, for a given CEARP instance defined on an undirected graph, an optimal solution exists in which no edge is traversed more than twice. This allows us to fix the large number M to 2 in F2 and to 1 in F3. Therefore, we use the dominance constraints of type (21) for the undirected instances. Note that, in this case, all variables in F3 are now binary. For the directed and mixed instances where M cannot be estimated effectively, the constraints (22) are used.

3.2. Comparisons of Lower Bounds

The first set of results compares the lower bounds obtained at the root node of our branch-and-cut algorithms. These are

TABLE 4. Comparison of performance of F2 and F3.

	F2			F3		
Data	Time	Gap	Result	Time	Gap	Result
ce500-1500-150-0.5-1	7202.29	4.21	106,474.2	7202.77	1.76	105,431.4
ce500-1500-150-0.5-2	7202.72	3.34	97,300.6	7203.01	2.13	97,335.3
ce500-1500-150-0.5-3	7202.46	0.82	112,463.3	7202.91	0.39	112,429.6
ce500-1500-150-0.5-4	7202.45	2.23	95,357.7	7202.98	2.53	95,378.1
ce500-1500-150-0.5-5	7202.51	2.35	102,403.2	7202.91	0.98	102,423.9
ce500-1500-150-1-1	7206.93	1.25	129,490.7	7207.52	0.40	129,459.7
ce500-1500-150-1-2	505.35	0	123,423.0	278.86	0	123,423.0
ce500-1500-150-1-3	121.73	0	133,418.3	598.91	0	133,418.3
ce500-1500-150-1-4	7206.53	2.11	116,431.6	7207.07	1.04	116,458.8
ce500-1500-150-1-5	7206.48	2.21	117,467.2	7206.98	1.55	117,403.4
ce500-1500-150-5-1	452.86	0	162,497.8	278.01	0	162,497.8
ce500-1500-150-5-2	128.71	0	160,492.7	101.33	0	160,492.7
ce500-1500-150-5-3	166.73	0	177,442.4	122.72	0	177,442.4
ce500-1500-150-5-4	64.19	0	151,452.9	78.69	0	151,452.9
ce500-1500-150-5-5	147.09	0	161,433.4	191.63	0	161,433.4
ce500-1500-150-10-1	152.53	0	174,404.1	174.82	0	174,404.1
ce500-1500-150-10-2	212.81	0	173,404.5	190.57	0	173,404.5
ce500-1500-150-10-3	197.64	0	185,430.8	201.41	0	185,430.8
ce500-1500-150-10-4	201.64	0	162,471.7	222.80	0	162,471.7
ce500-1500-150-10-5	945.25	0	168,434.3	240.02	0	168,434.3
ce500-1000-200-0.5-1	7200.69	1.59	76,333.0	4275.83	0	76,333.0
ce500-1000-200-0.5-2	7200.78	1.22	84,311.6	385.83	0	84,337.2
ce500-1000-200-0.5-3	7200.74	12.82	95,390.2	7200.95	0.80	89,353.5
ce500-1000-200-0.5-4	7200.71	2.51	76,326.2	7201.04	1.52	76,384.4
ce500-1000-200-0.5-5	137.05	0	85,397.0	1714.54	0	85,397.0
ce500-1000-200-1-1	461.58	0	82,387.1	4440.20	0	82,387.1
ce500-1000-200-1-2	140.30	0	89,396.5	257.57	0	89,396.5
ce500-1000-200-1-3	6209.50	0	98,351.8	279.85	0	98,351.8
ce500-1000-200-1-4	7202.02	1.95	82,382.3	7202.20	0.41	82,344.2
ce500-1000-200-1-5	38.03	0	91,315.6	59.10	0	91,315.6
ce500-1000-200-5-1	130.37	0	100,495.3	1233.24	0	100,495.3
ce500-1000-200-5-2	36.33	0	109,418.5	92.57	0	109,418.5
ce500-1000-200-5-3	121.71	0	114,462.3	121.69	0	114,462.3
ce500-1000-200-5-4	57.89	0	103,490.9	75.01	0	103,470.9
ce500-1000-200-5-5	19.58	0	112,425.0	52.97	0	112,425.0
ce500-1000-200-10-1	67.97	0	110,427.9	119.54	0	110,427.9
ce500-1000-200-10-2	57.21	0	113,494.1	83.24	0	113,494.1
ce500-1000-200-10-3	82.84	0	126,459.9	95.27	0	123,433.9
ce500-1000-200-10-4	69.35	0	115,481.4	54.81	0	115,481.4
ce500-1000-200-10-5	44.56	0	128,463.2	58.86	0	128,463.2

based on the formulations F1–F3 and the formulation similar to that in [6] in which all the CPLEX cuts except the zero-half cuts are turned on. The formulations are tested on the directed-graph instances of [5]. The results are given in Table 2. This table presents, for each formulation F1, F2, and F3, the lower bound obtained at the root node (lb_i). The value lb_4 is the lower bound obtained by the branch-and-cut algorithm of [6]. The best results are in bold.

The results shown in Table 2 indicate that the lb_4 bounds are always worse than the lb_2 bounds. This proves the efficiency of the addition of zero-half cuts in CPLEX. The results also imply that the formulations F2 and F3 are more efficient than F1. It seems that F3 is slightly better than F2, since lb_3 is larger than lb_2 in 30 of the 40 instances.

3.3. Overall Comparisons

Table 3 compares the five exact algorithms for the CEARP in terms of the number of instances successfully solved to optimality. Column F0 gives the results for the cutting-plane method introduced in [5], and columns F1, F2, and F3 give the results for our three branch-and-cut algorithms based on the corresponding formulations. Column F4 presents the number of successful instances for the algorithm proposed in [6] using CPLEX 11.2 instead of 9.1 which contains, among other improvements, the zero-half cuts. Again, the formulations are tested on the directed-graph instances of [5].

From Table 3, an interesting observation is that the branch-and-cut algorithms based on F1 and proposed in [6] are worse than the cutting-plane method introduced in [5]. The

TABLE 5. Details of branch-and-cut algorithms for F2 and F3.

	F2			F3			
Data	zero	user	bb	domi	zero	user	bb
ce500-1500-150-0.5-1	259	19,313	5817	3293	1910	9334	61,303
ce500-1500-150-0.5-2	263	20,336	6236	3575	2374	3701	113,437
ce500-1500-150-0.5-3	390	9797	16,380	2998	2785	1818	169,409
ce500-1500-150-0.5-4	592	7189	30,383	3469	2658	4539	71,336
ce500-1500-150-0.5-5	349	17,351	12,356	3333	2426	4453	66,382
ce500-1500-150-1-1	269	16,305	11,390	2797	2567	1540	264,417
ce500-1500-150-1-2	686	2015	16,371	2644	1569	361	20,352
ce500-1500-150-1-3	427	1229	3969	2900	2053	1300	40,377
ce500-1500-150-1-4	418	7696	22,392	2791	2101	2778	125,414
ce500-1500-150-1-5	262	15,310	12,392	3044	2029	6288	48,332
ce500-1500-150-5-1	355	3663	3119	2243	1140	264	33,310
ce500-1500-150-5-2	340	535	1270	2293	664	177	1561
ce500-1500-150-5-3	246	2040	343	2444	467	328	716
ce500-1500-150-5-4	231	158	136	2159	529	124	1047
ce500-1500-150-5-5	366	224	582	2226	984	149	4817
ce500-1500-150-10-1	242	164	174	2152	400	46	144
ce500-1500-150-10-2	289	1013	1286	2425	667	135	1158
ce500-1500-150-10-3	244	512	141	2259	376	448	487
ce500-1500-150-10-4	301	211	292	2396	806	92	2667
ce500-1500-150-10-5	639	6493	5309	2152	948	129	2741
ce500-1000-200-0.5-1	200	17,337	11,374	2110	3179	2845	180,430
ce500-1000-200-0.5-2	160	27,332	8960	1972	1618	1064	26,306
ce500-1000-200-0.5-3	116	37,316	1143	1942	4397	2381	194,481
ce500-1000-200-0.5-4	196	18,365	21,352	1989	1572	8734	64,379
ce500-1000-200-0.5-5	218	3417	1146	1994	3372	2458	51,354
ce500-1000-200-1-1	272	6291	4822	1666	2106	6997	104,488
ce500-1000-200-1-2	338	2709	3206	1590	1668	1232	10,394
ce500-1000-200-1-3	148	29,335	2314	1520	1613	738	18,344
ce500-1000-200-1-4	230	19,386	17,346	1444	681	8147	79,327
ce500-1000-200-1-5	160	945	70	1456	636	495	3987
ce500-1000-200-5-1	214	2670	785	1146	2354	2546	64,371
ce500-1000-200-5-2	182	221	125	1245	527	162	1329
ce500-1000-200-5-3	164	2870	285	1024	652	1656	3312
ce500-1000-200-5-4	161	1376	176	1070	385	137	266
ce500-1000-200-5-5	142	156	27	1208	339	264	194
ce500-1000-200-10-1	162	298	46	1020	568	257	2052
ce500-1000-200-10-2	273	810	194	1306	812	116	5612
ce500-1000-200-10-3	146	979	305	1067	421	733	1549
ce500-1000-200-10-4	133	961	79	1104	224	21	18
ce500-1000-200-10-5	134	192	27	1191	449	107	468

branch-and-cut algorithms based on F2 and F3 outperform the others. For $r = 150$ m, it seems that F2 and F3 are equivalent. For $r = 200$ m, F3 is better because it can solve 2 instances with $t = 0.5$ that F2 can not. Therefore, in the next subsection, we compare only F2 and F3 on other criteria and other instances.

3.4. Detailed Comparisons of Directed-Graph Instances

This subsection provides the results for the branch-and-cut algorithms using F2 and F3. For each formulation, Table 4 shows the time required (time), the gap (gap), and the solution value (result), and Table 5 presents the number of dominance constraints of type (22) (*domi*), the number of zero-half cuts (*zero*), the number of connectivity cuts (*user*), and the number of vertices (*bb*) in the branch-and-bound tree. The gap for F3 is better than that for F2. F3 also finds better solutions than F2. We believe this is because F2 contains Big-M constraints that lead to numerical difficulties because of the many generated connectivity constraints (the *user* column in Table 5). These increase the size of the model, and so CPLEX needs more time to process each node. Therefore, there are less chances to find good solutions. In contrast, the branch-and-bound tree of F3 is much larger. This is because we use the constraints (22) instead of the constraints (21) to reduce the size of the search tree and it seems that the constraints (21) are more efficient than the constraints (22).

3.5. Comparisons of Undirected- and Mixed-Graph Instances

Table 6 gives the results for the two formulations on undirected-graph instances. F3 can solve the instance MADR-7-3 that F2 can not. It is also faster on 22 of 30

TABLE 6. Comparison of performance of F2 and F3 on undirected instances.

	F2					F3				
Data	lb1	Gap	bb	Time	Result	lb1	Gap	bb	Time	Result
ALBA-3-1	2391.8	0	236	5.34	2511	2319.2	0	209	2.08	2511
ALBA-3-2	2124.7	0	119	4.14	2324	2011.2	0	74	2.15	2324
ALBA-3-3	2012.8	0	74	1.88	2155	2070.7	0	30	1.32	2155
ALBA-3-4	2363.7	0	959	53.42	3074	2621.0	0	334	13.39	3074
ALBA-3-5	2244.0	0	71	3.63	2440	2355.1	0	21	2.04	2440
ALBA-5-1	2845.0	0	703	28.15	3125	2712.3	0	1105	24.26	3125
ALBA-5-2	2468.3	0	637	24.70	2926	2480.5	0	1655	35.78	2926
ALBA-5-3	3013.8	0	279	9.93	3170	2961.5	0	446	13.12	3170
ALBA-5-4	2447.8	0	298	8.78	2584	2405.8	0	433	7.77	2584
ALBA-5-5	2472.4	0	477	24.66	2642	2535.6	0	112	3.22	2642
ALBA-7-1	2989.7	0	495	21.33	3397	2879.4	0	528	13.64	3397
ALBA-7-2	3200.1	0	793	29.71	3558	3192.1	0	740	21.77	3558
ALBA-7-3	3234.6	0	1939	68.88	3647	3125.3	0	1913	64.31	3647
ALBA-7-4	3255.8	0	365	15.20	3461	3264.4	0	789	24.63	3461
ALBA-7-5	2540.8	0	401	23.27	2821	2642.5	0	249	8.81	2821
MADR-3-1	2511.2	0	11,315	2027.39	2925	2658.0	0	4757	137.62	2925
MADR-3-2	3103.3	0	10,353	1972.94	3665	3197.5	0	28,355	2731.95	3665
MADR-3-3	2580.7	0	2556	366.50	3045	2714.9	0	2516	112.25	3045
MADR-3-4	2829.3	0	8838	2463.51	3295	2936.4	0	11,386	794.31	3295
MADR-3-5	2704.7	0	3603	552.06	3165	2760.4	0	7960	1303.2	3165
MADR-5-1	3479.6	0	7116	1398.53	3945	3568.7	0	4261	593.94	3945
MADR-5-2	4234.9	0	3124	280.35	4570	4239.4	0	6871	213.86	4570
MADR-5-3	3850.6	0	17,362	4067.07	4505	3810.2	0	4395	480.13	4505
MADR-5-4	3780.9	0	1526	184.09	4020	3840.0	0	2605	180.52	4120
MADR-5-5	3583.5	0	7894	1156.25	4010	3695.5	0	6199	738.96	4010
MADR-7-1	4287.3	0	5041	850.95	4645	4329.4	0	14,385	2512.30	4645
MADR-7-2	4363.8	0	7424	1499.57	4650	4350.3	0	4825	477.29	4650
MADR-7-3	4120.9	1.05	33,374	7200.08	4620	4234.8	0	16,384	2128.87	4620
MADR-7-4	4190.7	2.27	27,365	7200.07	4655	4186.7	2.48	39,364	7200.06	4645
MADR-7-5	4370.1	0	12,306	2958.65	4735	4338.2	0	23,333	4535.04	4735

instances. Therefore, on undirected instances F3 outperforms F2. Note that in this case, all the variables of F3 are binary and this is probably why F3 is better than F2.

Table 7 gives the results for mixed-graph instances. All the results are averages over five instances. F2 can solve more instances than F3 but once again its gap is, in some cases, poor.

3.6. Results for Upper Bounds

Tables 8 and 9 give the results for algorithms UB1 and UB2. In these tables, M_{CPP} is the value of M calculated by solving the CPP problems. The names of instances in bold indicate that these instances were proved optimal by the branch-and-cut algorithms. The BnC Time column gives the running time in seconds of the branch-and-cut algorithm based on F2. The Gap column displays the gap in percent of upper bounds to the best solutions found by the two branch-and-cut algorithms in the previous subsection. The negative results imply that the upper-bound algorithms found better solutions than the branch-and-cut algorithms. From the results of the exact algorithms, we observe that an arc is rarely crossed more than five times in the solution. Therefore, we

TABLE 7. Comparison of performance of F2 and F3 on mixed-graph instances.

	F2					F3				
Data	succ	Gap	bb	Time	Result	succ	Gap	bb	Time	Result
MB0537-0.5	4	1.75	10,330.2	3379.42	17,392.0	0	1.07	38,379.6	7200.72	17,360.4
MB0537-1	3	0.59	10,325.2	4707.06	18,346.8	0	1.33	63,397.2	7201.54	18,371.4
MB0537-5	5	0	3242.0	185.89	21,312.2	3	0.19	138,495.6	3937.83	21,312.2
MB0537-10	5	0	1040.2	74.01	22,366.2	5	0	163,433.6	2418.27	22,366.2
MB0547-0.5	0	9.28	5450.2	7200.63	15,359.8	0	6.95	19,357.4	7201.09	15,389.2
MB0547-1	0	3.38	14,369	7201.74	17,355.0	0	3.48	42,325.4	7202.20	17,316.2
MB0547-5	5	0	4428.6	470.27	21,343.2	4	0.13	27,333.8	1639.97	21,343.2
MB0547-10	5	0	1461.6	79.97	22,304.0	5	0	31,392.4	756.16	22,304.0

TABLE 8. Upper bounds based on F2 for directed-graph instances.

	UB1		UB2-F2, $M = 5$		UB2-F2, $M = M_{CPP}$			Bnc
Data	Time	Gap	Time	Gap	M_{CPP}	Time	Gap	Time
ce500-1500-150-0.5-1	6.93	14.71	3819.53	−0.32	11	1128.79	−0.32	7202.29
ce500-1500-150-0.5-2	7.49	16.97	473.07	−0.69	12	1084.34	−0.69	7202.72
ce500-1500-150-0.5-3	7.80	15.81	141.14	−0.02	15	171.44	−0.02	7202.46
ce500-1500-150-0.5-4	8.28	13.30	1337.99	−0.80	12	7202.44	−0.79	7202.45
ce500-1500-150-0.5-5	7.30	13.97	1699.12	−0.33	10	7202.37	−0.33	7202.51
ce500-1500-150-1-1	10.97	10.02	143.32	−0.07	11	129.40	−0.07	7206.93
ce500-1500-150-1-2	11.25	6.24	83.17	0	12	90.92	0	505.35
ce500-1500-150-1-3	11.50	6.39	153.90	0	15	121.17	0	121.73
ce500-1500-150-1-4	11.60	12.62	480.28	−0.10	12	746.87	−0.10	7206.53
ce500-1500-150-1-5	11.19	11.38	7206.52	−0.12	10	7206.26	0.03	7206.48
ce500-1500-150-5-1	57.61	2.94	140.66	0	11	117.08	0	452.86
ce500-1500-150-5-2	54.70	2.41	133.76	0	12	132.17	0	128.71
ce500-1500-150-5-3	54.15	3.58			15	147.08	0	166.73
ce500-1500-150-5-4	51.24	3.62	93.06	0	12	79.82	0	64.19
ce500-1500-150-5-5	53.93	3.04	147.37	0	10	166.00	0	147.09
ce500-1500-150-10-1	119.36	1.75	143.05	0	11	152.90	0	152.53
ce500-1500-150-10-2	107.62	2.26	164.46	0	12	175.16	0	212.81
ce500-1500-150-10-3	118.05	2.57			15	178.73	0	197.64
ce500-1500-150-10-4	101.88	2.78	175.58	0	12	178.49	0	201.64
ce500-1500-150-10-5	115.54	2.91	250.78	0	10	218.16	0	945.25
ce500-1000-200-0.5-1	4.58	30.25	88.65	0	10	54.29	0	7200.69
ce500-1000-200-0.5-2	4.69	42.00	66.63	0	14	161.62	0	7200.78
ce500-1000-200-0.5-3	4.62	33.06	96.97	0	12	354.75	0	7200.74
ce500-1000-200-0.5-4	4.57	41.92	287.06	−0.17	12	252.59	−0.17	7200.71
ce500-1000-200-0.5-5	4.68	39.59	89.17	0	11	81.70	0	137.05
ce500-1000-200-1-1	5.78	24.22	78.95	0	10	106.86	0	461.58
ce500-1000-200-1-2	5.87	35.84	73.87	0	14	182.10	0	140.30
ce500-1000-200-1-3	6.07	23.36	180.66	0	12	81.72	0	6209.50
ce500-1000-200-1-4	5.94	30.50	112.85	0	12	358.85	0	7202.02
ce500-1000-200-1-5	5.99	28.21	23.47	0	11	15.51	0	38.03
ce500-1000-200-5-1	19.57	30.18	60.29	0	10	97.54	0	130.37
ce500-1000-200-5-2	17.98	10.98	74.72	0	14	81.14	0	36.33
ce500-1000-200-5-3	21.04	10.41	58.55	0	12	71.51	0	121.71
ce500-1000-200-5-4	21.14	18.55	34.15	0	12	42.40	0	57.89
ce500-1000-200-5-5	21.19	18.32	24.90	0	11	28.32	0	19.58
ce500-1000-200-10-1	39.85	9.79	67.88	0	10	95.28	0	67.97
ce500-1000-200-10-2	35.85	6.63	49.28	0	14	55.72	0	57.21
ce500-1000-200-10-3	42.70	4.45	80.61	0	12	86.33	0	82.84
ce500-1000-200-10-4	42.54	11.91	49.51	0	12	46.20	0	69.35
ce500-1000-200-10-5	43.33	12.98	62.41	0	11	57.18	0	444.56

also test the MIP-based constructive algorithms with $M = 5$. Note that this algorithm is not competitive on undirected-graph instances, because in these cases, we can bound the large number M efficiently. When M is bounded more strictly, F2 is much better than F3, so we do not present the results of the algorithm for F3.

The results for directed- and mixed-graph instances show the good performance of the algorithm UB2. It not only finds better solutions but is also faster in almost all the instances that the branch-and-cut algorithms can not solve exactly. The performance of algorithm with $M = 5$ is slightly better than with $M = M_{CPP}$ but bounding M too strictly can make the problem infeasible, two instances ce500-1500-5-3-150 and ce500-1500-10-3-150 for example. The quality of UB1 is poor, especially on the directed-graph instances where $r = 200$ m as well as on the mixed-graph instances. It only works better on the easy instances with $r = 150$ m and $t = 0.5, 1$. However, it is much faster than UB2.

4. CONCLUSIONS

We have proposed a new formulation for the CEARP. In contrast to two formulations in the literature, this formulation has an important advantage: it does not require Big-M constraints. Branch-and-cut algorithms have been developed for three formulations, and the three formulations have been compared. The results show that the branch-and-cut algorithms based on our new formulation and on the formulation of [6] outperform the other algorithms considered. In comparison with the formulation of [6], our new formulation is better on directed- and undirected-graph instances but worse on mixed-graph instances. Normally, our new formulation

TABLE 9. Upper bounds based on F2 for mixed-graph instances.

	UB1		UB2-F2, $M = 5$		UB2-F2, $M = M_{CPP}$			BnC
Data	Time	Gap	Time	Gap	M_{CPP}	Time	Gap	Time
MB0537-0.5-1	4.77	34.40	345.39	0	7	278.38	0	992.57
MB0537-0.5-2	4.77	39.41	382.67	0	7	397.07	0	4491.65
MB0537-0.5-3	5.04	36.08	2726.26	−0.51	7	981.18	−0.51	7200.69
MB0537-0.5-4	4.73	37.74	146.45	0	7	212.68	0	690.54
MB0537-0.5-5	5.09	31.65	478.33	0	7	214.05	0	3521.67
MB0537-1-1	5.79	33.56	276.76	0	7	155.04	0	1382.62
MB0537-1-2	5.79	33.60	108.31	0	7	127.44	0	3629.55
MB0537-1-3	5.78	33.08	393.87	−0.45	7	287.98	−0.45	7201.13
MB0537-1-4	5.53	38.33	723.45	0	7	3143.52	0	7201.26
MB0537-1-5	5.88	34.87	2471.76	0	7	617.78	0	4120.76
MB0537-5-1	13.40	17.44	274.46	0	7	283.69	0	593.70
MB0537-5-2	13.08	15.59	41.81	0	7	50.01	0	60.67
MB0537-5-3	13.78	18.12	92.07	0	7	96.55	0	138.32
MB0537-5-4	14.10	17.96	63.01	0	7	107.87	0	93.19
MB0537-5-5	13.66	15.07	64.30	0	7	57.08	0	43.58
MB0537-10-1	23.27	13.45	69.48	0	7	72.61	0	73.15
MB0537-10-2	25.68	14.77	67.03	0	7	74.79	0	75.52
MB0537-10-3	25.64	18.29	185.10	0	7	144.82	0	114.87
MB0537-10-4	25.22	11.42	49.43	0	7	30.24	0	30.61
MB0537-10-5	24.12	14.85	40.86	0	7	46.35	0	75.92
MB0547-0.5-1	5.60	44.89	7200.58	−4.03	14	7200.88	−1.40	7200.65
MB0547-0.5-2	5.23	41.12	1460.73	−1.20	14	2645.67	−1.20	7200.60
MB0547-0.5-3	5.22	45.92	7200.57	−1.06	14	2215.78	−1.33	7200.59
MB0547-0.5-4	5.31	39.19	7200.69	−1.34	14	7201.31	−0.65	7200.62
MB0547-0.5-5	5.21	40.64	7200.58	−1.65	14	7200.61	−1.63	7200.68
MB0547-1-1	6.25	36.72	66.54	−0.17	14	233.73	−0.17	7201.69
MB0547-1-2	6.27	33.26	383.36	−0.31	14	1211.10	−0.31	7201.86
MB0547-1-3	6.54	26.39	7201.62	−1.06	14	7201.70	−1.09	7201.60
MB0547-1-4	6.20	32.59	954.23	−0.21	14	969.81	−0.21	7201.87
MB0547-1-5	6.21	33.52	7201.62	−0.69	14	7201.59	−0.66	7201.67
MB0547-5-1	17.35	29.24	17.54	0	14	24.49	0	46.89
MB0547-5-2	17.62	40.39	18.28	0	14	43.18	0	32.89
MB0547-5-3	18.28	139.02	18.05	0	14	67.34	0	2158.19
MB0547-5-4	17.27	14.59	33.64	0	14	50.53	0	40.31
MB0547-5-5	17.46	16.91	64.69	0	14	52.02	0	73.08
MB0547-10-1	22.94	44.51	17.99	0	14	48.91	0	98.00
MB0547-10-2	35.85	14.71	60.13	0	14	67.20	0	54.33
MB0547-10-3	32.65	12.50	83.55	0	14	105.39	0	138.32
MB0547-10-4	31.59	11.56	69.47	0	14	64.96	0	47.86
MB0547-10-5	31.88	14.39	55.85	0	14	67.30	0	61.35

gives a better gap, but an enormous branch-and-bound tree is the price to pay. We also propose a MIP-based constructive heuristic for the CEARP on directed- and mixed-graph instances based on the formulation of [6] in which the large number M is bounded more strictly.

REFERENCES

[1] N. Christofides, V. Campos, A. Corberán, and E. Mota, An algorithm for the rural postman problem on a directed graph, Math Program Stud 26 (1986), 155–166.

[2] M.O. Ball and M.J. Magazine, Sequencing of insertions in printed circuit board assembly, Oper Res 36 (1988), 192–201.

[3] V. Campos and J.V. Savall, A computational study of several heuristics for the directed rural postman problem, Comput Optim Appl 4 (1993), 67–77.

[4] B. Golden, S. Raghavan, and E. Wasil, "Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network," The vehicle routing problem: Lastest avances and new challenges, Springer-Verlag, Berlin, 2008, pp. 487–501.

[5] M-H. Ha, N. Bostel, A. Langevin, and L-M. Rouseau, "An exact algorithm for close enough traveling salesman problem," Proceedings of the 1st International Conference on Operations Research and Enterprise Systems, C-J. Luz and F. Valente (Editors), Vilamoura, Algarve, Portugal, February 2012, pp. 233–238, SciTePress 2012 ISBN 978-989-8425-97-3.

[6] M. Drexl, On some generalized routing problems. Ph.D. dissertation, Rheinisch-Westfalishe Technische Hochschule Aachen, Aachen, Germany, 2003.

[7] M. Gendreau, G. Laporte, and F. Semet, The covering tour problem, Oper Res 45 (1997), 568–576.

[8] R. Baldacci, M.A. Boschetti, V. Maniezzo, and M. Zamboni, "Scatter search methods for covering tour problem," Metaheuristic optimization via memory and evolution, R. Sharda, S. Voss, C. Rego, and B. Alidaee (Editors), Springer Verlag, Berlin, 2005, pp. 55–91.

[9] M. Hachicha, M.J. Hodgson, G. Laporte, and F. Semet, Heuristics for the multi-vehicle covering tour problem, Comput Oper Res 27 (2000), 29–42.

[10] D. Gulczynski, J. Heath, and C. Price, "The close enough traveling salesman problem: A discussion of several heuristics," Perspectives in operations research: Papers in honor of saul Gass' 80th birthday, F. Alt, M. Fu, and B. Golden (Editors), Springer Verlag, Berlin, 2006, pp. 271–283.

[11] J. Dong, N. Yang, and M. Chen, "Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem," Extending the horizons: Advances in computing, optimisation, and decision technologies, E. Baker, A. Joseph, A. Medrotra, and M. Trick (Editors), Springer Verlag, Berlin, 2007, pp. 145–163.

[12] E. Balas and S.M. Ng, On the set covering polytope: All the facets with coefficients in 0, 1, 2, Math Program 43 (1986), 57–69.

[13] M. Sánchez-García, M.I. Sobrón, and B. Vitoriano, On the set covering polytope: Facets with coefficients in {0, 1, 2, 3}, Ann Oper Res 81 (1998), 343–356.

[14] A. Corberán, E. Mota, and J.M. Sanchis, A comparison of two different formulations for arc routing problems on mixed graphs, Comput Oper Res 33 (2006), 3384–3402.

[15] A. Corberán, A.N. Letchford, and J.M. Sanchis, A cutting plane algorithm for the general routing problem, Math Program 90 (2001), 291–316.

[16] E. Benavent, A. Carrotta, A. Corberán, J.M. Sanchis, and D. Vigo, Lower bounds and heuristics for the windy rural postman problem, Eur J Oper Res (2007), 855–869.

[17] G. Ghiani and G. Laporte, A branch-and-cut algorithm for the undirected rural postman problem, Math Program 87 (2000), 467–481.