

The covering Chinese postman problem: exact and heuristic methods

Rafael Kendy Arakaki

Instituto de Computação - Universidade Estadual de Campinas
Av. Albert Einstein 1251, 13083-852, Campinas, SP
rafael.arakaki@ic.unicamp.br

Fábio Luiz Usberti

Instituto de Computação - Universidade Estadual de Campinas
Av. Albert Einstein 1251, 13083-852, Campinas, SP
fusberty@ic.unicamp.br

ABSTRACT

The covering Chinese postman problem (CCPP) is a combinatorial optimization problem. Given an undirected graph with cost and covering functions associated to the edges, this problem aim to find a minimum cost tour that covers all the edges of the graph. The problem is modeled by an integer programming mathematical formulation. An exact branch-and-cut method and a heuristic method are proposed for the CCPP. Extensive computational experiments were performed. Branch-and-cut method was able to obtain 325 optimal solutions out of 390 instances, while heuristic method obtained better solutions for some large instances in shorter processing times.

KEYWORDS. Covering Chinese postman problem, arc routing, branch-and-cut, combinatorial optimization, heuristic.

OC (Combinatorial Optimization), MP (Mathematical Programming)

1. Introduction

Node routing problems, such as the well known *traveling salesman problem* (TSP), aim to find optimum routes while visiting a set of nodes of a network. On the other hand, arc routing problems aim to determine optimum routes that visit some arcs of the graph, with one or more side constraints [Eiselt et al., 1995].

The *covering Chinese postman problem* (CCPP) is an arc routing problem with covering constraints whose solution consist of a minimum cost tour that must visit a set of arcs which covers all the edges of a network. The *covering salesman problem* (CSP) [Current and Schilling, 1989], the node routing counterpart of CCPP, has been object of research in the last decades [Current and Schilling, 1989; Salari and Naji-Azimi, 2012; Ozbaygin et al., 2016]. To our knowledge, the CCPP is a new proposed problem.

Our contribution. This paper proposes an *integer programming* (IP) formulation for the CCPP and a heuristic method called *greedy randomized heuristic* (GRH). The IP formulation is solved by an exact branch-and-cut method. Extensive computational experiments were performed to evaluate the proposed methods. Results show that the branch-and-cut was able to obtain 325 optimum solutions out of 390 instances considered in the experiments. Also, the GRH was found to obtain better solutions than branch-and-cut method for 24 large instances while spending much less computational time.

Section 2 provides the literature background. Section 3 formally defines the CCPP. Section 4 describe an IP formulation for the problem. In Section 5, a heuristic method is proposed. Section 6 contains the computational experiments. Finally, Section 7 provides the final comments.

2. Literature review

The *Chinese postman problem* (CPP) [Eiselt et al., 1995] is defined on a undirected graph $G(V, E)$ with a cost function associated to the edges and the objective is to find the minimum cost closed circuit (it ends and starts in the same node) that visit each edge $(u, v) \in E$ at least once. This problem can be solved in polynomial time using an optimal matching algorithm [Eiselt et al., 1995]. Many variations of the CPP, however, are NP-hard problems, such as the mixed CPP [Minieka, 1979] and windy CPP [Pearn and Li, 1994].

Current and Schilling [1989] proposed a node routing problem with covering constraints: the *covering salesman problem* (CSP), which is a generalization of the well known traveling salesman problem. In this problem, the objective is to find a minimum cost tour of a subset of cities such that every city not visited by the tour is within some predetermined fixed covering distance of a city that is on the tour. Some papers have discussed on the applications of this problem and some of its generalizations in the fields of emergency management and disaster planning [Altay and Green III, 2006; Reina et al., 2013; Salari and Naji-Azimi, 2012].

Drex1 [2014] proposed the *generalized directed rural postman problem* (GDRPP) which objective is to find the minimum cost tour in a digraph such that the arcs traversed must cover all the predefined subsets of arcs of the digraph. More specifically, at least one arc of each predefined subset must be visited by the tour. The subsets need neither be a partition of the arc set, nor need they be disjoint. Drex1 demonstrated how to transform many arc routing problems to the GDRPP, such as: mixed Chinese postman problem, arc routing with turn penalties, arc routing with zigzag service, among others. As solution method, the author proposed a branch-and-cut algorithm which was able to obtain very good results including many optimum solutions for the considered instances.

3. Definition

The covering Chinese postman problem (CCPP) is a new proposed variant of the CPP that considers covering constraints. The CCPP can be formally defined as next. Let $G(V, E)$ be

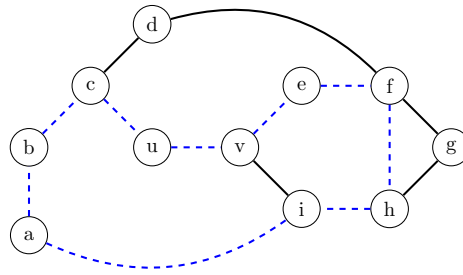


Figure 1: Example of a CCPP instance and its solution.

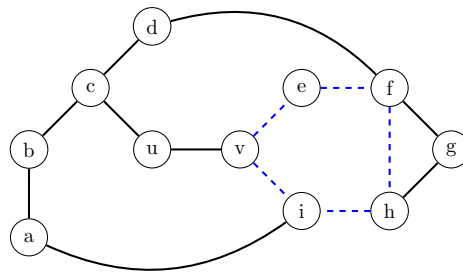


Figure 2: Example of a CCPP instance with a larger covering function and its solution.

an undirected graph and $c(u, v) \geq 0$ be a non-negative cost associated to each edge $(u, v) \in E$. Consider a covering function $D(u, v)$ associated to each edge $(u, v) \in E$ that represents the set of edges covered by (u, v) . Also, it is required that for each $(u, v) \in E$: (i) $(u, v) \in D(u, v)$; and (ii) $(p, q) \in D(u, v) \iff (u, v) \in D(p, q)$. The objective is to find a minimum-cost closed circuit such that all the edges in G are covered by at least one of the edges visited by the circuit.

An example of solution for CCPP is given by Figure 1 where all edges have one unit of cost and the covering function for each edge $(u, v) \in E$ is defined as the set of edges that are incident to either the nodes u or v . Figure 2 shows a solution for the same graph and costs but defining the covering function of each edge $(u, v) \in E$ as all the edges whose both extreme points are at most from three units of distance from both u and v .

4. Mathematical Model

An integer programming mathematical formulation for the CCPP is proposed next. Although the graph G is undirected, the variables in the model are directed. Also, although a CCPP solution can visit an specific edge many times, the variables in the model are binary. In fact, it can be showed that exist at least one optimal solution for the CCPP that does not visit the same edge $(u, v) \in E$ more than twice and, if it visits (u, v) twice, then it is in opposite directions $((u, v)$ and $(v, u))$. Welz [1994] proved that property for the *capacitated arc routing problem* but the proof can be also applied to CCPP. Therefore, the model works on $2|E|$ binary variables in the form of directed arcs x_{ij}, x_{ji} representing each edge $(i, j) \in E$ visited in opposite directions.

$$(CCPP)$$

$$MIN \sum_{(i,j) \in E} c_{ij}(x_{ij} + x_{ji}) \quad (1)$$

s.t.

$$\sum_{j \in \delta(i)} x_{ij} = \sum_{j \in \delta(i)} x_{ji} \quad (i \in V) \quad (2)$$

$$\sum_{(i,j) \in D(u,v)} (x_{ij} + x_{ji}) \geq 1 \quad ((u,v) \in E) \quad (3)$$

$$\sum_{(i,j) \in (S, \tilde{S})} (x_{ij} + x_{ji}) \geq 2(x_{kl} + x_{mn} - 1)$$

$$(\forall S \subset V, \forall (k,l), (m,n) \in E : k,l \in S \text{ and } m,n \in \tilde{S}) \quad (4)$$

$$x_{ij}, x_{ji} \in \{0, 1\}, \forall (i,j) \in E \quad (5)$$

$\delta(i)$: edges incident to node $i \in V$.

In this model, x_{ij} and x_{ji} assume value 1 if and only if the edge $(i,j) \in E$ is visited in each direction. Therefore, the solution cost of the closed circuit is given by objective function (1). Constraints (2) are degree constraints. Constraints (3) force that the solution must visit at least one edge of the covering functions associated to each edge. The connectivity constraints (4) force that every pair of visited edges must be connected in the solution. More specifically, they state that if two edges are selected, respectively belonging to (S, S) and $(\tilde{S}, \tilde{S})$, then at least two cut edges in $(S, \tilde{S})$ must be selected as well, otherwise the solution is not connected. Constraints (5) define the domain of variables x .

In terms of implementation, careful must be taken in relation to constraints (4) since they are of exponential number. A complete enumeration of them is only possible for very small instances. Inspired by the work of Belenguer and Benavent [2002], the solution adopted was to add incrementally the constraints (4) as they are needed in a *branch-and-cut* fashion. The branch-and-cut algorithm iteratively adds a subset of the exponential sized set of constraints (4) that are identified by a *separation algorithm* (Section 4.1) developed specifically for this formulation. When no violated constraints are found by the separation algorithm and the solution is integer, then that solution is feasible. Details of how a branch-and-cut algorithm operates would be omitted because of complexity, while an interested reader can find it in [Jünger et al., 2009].

4.1. Separation algorithm for connectivity constraints

Since connectivity constraints are of exponential number it is impractical to solve a model containing all of them, except for very small instances. Therefore, a polynomial-time separation algorithm was developed to identify violated connectivity constraints (4) on demand and add them to the model. Given a candidate solution x^* for CCPP that obeys constraints (2) and (3), the separation algorithm identifies a subset of connectivity constraints (4) that are violated by x^* . The method works on both integer and real (relaxed integer) variables. The procedure pseudocode is described in Algorithm 1.

Given candidate solution x^* , the separation algorithm start by considering a graph $G^*(V, E^*)$ induced by the edges $E^* = \{(i,j) \in E : x_{ij}^* + x_{ji}^* \geq \delta\}$ for a small $0 < \delta \ll 1$. From that, the

Gomory-Hu algorithm [Gomory and Hu, 1961] is executed on G^* by considering the capacity of each edge $e \in E^*$ as the sum of corresponding arc variables: $x_{ij}^* + x_{ji}^*$. The Gomory-Hu algorithm computes the minimum flow cut edges for each pair of nodes $u, v \in V$. For each found minimum flow cut, the separation algorithm verifies if there is a suitable connectivity constraint (4) violated, in other words, for a given set S it searches for two edges: an edge $(k, l) \in (S, S)$ and an edge $(m, n) \in (\tilde{S}, \tilde{S})$ such that the minimum flow cut value is less than $2(x_{kl} + x_{mn} - 1)$. The algorithm output is a set of connectivity constraints, corresponding to the minimum flow cuts, that are violated by x^* .

Data: Given solution x^* for CCPP that obeys constraints (2) and (3).

Result: A set VC of violated connectivity constraints, if there is any.

Consider a graph $G^*(V, E^*)$ induced by set of edges $E^* = \{(i, j) \in E : x_{ij}^* + x_{ji}^* \geq \delta\}$;

Execute Gomory-Hu algorithm on G^* considering the flow of each $(i, j) \in E^*$ as $x_{ij}^* + x_{ji}^*$;

for each min (flow) cut S found by Gomory-Hu algorithm **do**

$(k, l) \leftarrow$ edge such that $x_{kl}^* \geq x_{ij}^*, \forall (i, j) \in (S, S)$;

$(m, n) \leftarrow$ edge such that $x_{mn}^* \geq x_{ij}^*, \forall (i, j) \in (\tilde{S}, \tilde{S})$;

if Constraint (4) is violated by $S, x^*, (k, l)$ and (m, n) **then**

$v \leftarrow$ violated constraint by $S, (k, l)$ and (m, n) ;

$VC \leftarrow VC \cup v$;

end

end

return VC ;

Algorithm 1: Separation algorithm for connectivity constraints.

5. Heuristic method

A heuristic method called *greedy randomized heuristic* (GRH) is proposed for the CCPP. The primary objective of GRH is to generate feasible solutions with reasonable quality for large instances. In addition, GRH intends to consume much less computing time than the exact method (Section 4). Our heuristic was inspired by the GRASP metaheuristic [Resende and Ribeiro, 2014], but different from the latter it does not contain any adaptive feature.

The GRH pseudocode is described in Algorithm 2. Given a graph $G(V, E)$, the covering function D and a parameter k of number of iterations, the algorithm is executed and the output is the best solution found. The GRH first performs a preprocessing procedure and then runs k independent iterations. Each GRH iteration produces one solution and consists of a constructive phase followed by a local search phase.

The preprocessing procedure starts computing a matrix $SP[u][v]$ representing the shortest path from node u to v for each $u, v \in V$. The matrix is computed using the Dijkstra's algorithm [Cormen et al., 2001]. From that, an ordered non-increasing list $CE(u)$ of closest edges, for each node $u \in V$ is computed. More specifically, the distance from node u to edge (i, j) is defined as: $\min(SP[u][i], SP[u][j])$, where SP is the matrix of shortest path costs computed earlier. This step is very important for the heuristic performance because the distance between two nodes SP and the list of closest edges CE are computed only once but are intensively used by all following iterations of GRH.

In the constructive phase a feasible solution for CCPP is generated from scratch. The solution is represented by a sequence of directed arcs $(u_1, v_1), (u_2, v_2), \dots, (u_n, v_n)$ which gives the tour direction and for each subsequent $v_i \neq u_{i+1}$ a shortest path from v_i to u_{i+1} is implicitly implied. The return from node v_n to u_1 is also implied in this arrangement. The constructive phase starts by selecting one edge (u, v) of E randomly (each edge has equal probability). The edge

(u, v) is added to the solution in a directed arc form (u then v) and all edges that are in the covering function $D(u, v)$ are marked. Then, the procedure iteratively selects the closest edge $(i, j) \in E$ to the last visited node v , considering the computed list $CE(v)$, and that is not covered by any edge in the solution. From that the edge (i, j) is added to the solution and all edges in $D(i, j)$ are marked. The process is repeated until all edges are covered (marked) by the edges added to the solution.

Data: Graph $G(V, E)$, covering function D , parameter k of number of iterations.

Result: The best solution produced by GRH in the k iterations.

// preprocessing phase

Compute $SP[u][v]$ for each $u, v \in V$;

Compute $CE(u)$ for each $u \in V$;

$bestSol \leftarrow \emptyset$;

for k iterations **do**

$sol \leftarrow$ a random edge $(u, v) \in E$;

$M \leftarrow D(u, v)$;

$ln \leftarrow v$;

// constructive phase

while $|M| < |E|$ **do**

Search in $CE(ln)$ the closest edge (i, j) such that $(i, j) \notin M$;

Add (i, j) to sol at the final position;

$M \leftarrow M \cup D(i, j)$;

$ln \leftarrow j$;

end

// local search phase

while sol has a improving *remove* **do**

sol is updated by the first improving *remove* found;

end

while sol has a improving *exchange* **do**

sol is updated by the first improving *exchange* found;

end

while sol has a improving *2-OPT* **do**

sol is updated by the first improving *2-OPT* found;

end

if $cost(sol) < cost(bestSol)$ or $bestSol = \emptyset$ **then**

$bestSol \leftarrow sol$;

end

end

return $bestSol$;

Algorithm 2: Greedy randomized heuristic (GRH) for CCP.

The local search phase starts from a feasible solution s and attempts to improve it by considering three sets of moves in order. If, starting from s , any move finds an improving feasible solution s' with lower cost, that move is immediately performed rather than examining all possible moves for a solution. The solution is updated and that set of moves is examined again. After examining a set of moves to exhaustion without finding any improvement, the next set of moves is examined. Three sets of moves are considered in order:

1. *Remove*: remove from the solution representation any arc (u, v) that can be removed without losing feasibility. In other words, all edges in $D(u, v)$ are covered by at least two edges represented in the solution.
2. *Exchange*: replace in the solution representation an arc (u, v) by an arc (p, q) such that the solution is still feasible and has a lower cost.

3. *2-OPT*: change the order which the edges in the solution are visited [Hertz et al., 1999]. Considering that any pair of subsequent edges in the solution are connected by shortest path links, the 2-OPT move removes two shortest path links and adds two others shortest path links such that it stills represents a closed tour.

The local search ends when the 2-OPT set of moves is examined and no improvement is found.

6. Computational experiments

Computational experiments were performed to compare the proposed exact branch-and-cut (Section 4) and heuristic GRH (Section 5) methods on benchmark instances from the literature. Experiments were performed in an Intel Core i7-6700K 4.0 GHz, 16 GB RAM and Linux 64-bit operating system. The methods were coded in C++ using the LEMON library for graph algorithms [Dezső et al., 2011] and compiled by g++ v4.8 with “-O3” flag. The branch-and-cut algorithm used the commercial IP solver Gurobi v8.1 with default parameters and time limit of 1800 seconds for each instance. Separation algorithm used $\delta = 0.01$ as tolerance constant. The GRH was executed for $k = 10000$ iterations.

The experiments employed 6 sets of benchmark instances that were originally designed for the *rural postman problem* (RPP): the “Alba” set consisting of two instances modeled from the road-network of the city of Albaida, Spain, from Corberán and Sanchis [1994]; the 24 computer-generated “P” instances of Christofides et al. [1981]; the 20 randomly generated “R” instances from Hertz et al. [1999]; the 36 instances with nodes degree four “D” and the 36 grid instances “G” both from Aráoz et al. [2009]; and 12 “UR500” large instances from Corberán et al. [2007].

For each RPP instance set, three different covering functions D_α for $\alpha \in \{1.0, 2.0, 3.0\}$ were considered. Therefore, from 6 original instance sets for RPP, a total of 18 instance sets for CCPP were generated.

The covering functions $D_\alpha(u, v)$ are defined for each $(u, v) \in E$ as the set of edges $(i, j) \in E$ such that: $\max(SP[i][u], SP[i][v], SP[j][u], SP[j][v]) \leq \alpha \times \mu(c(E))$, where $SP[x][y]$ is the shortest path distance from node x to node y , $\mu(c(E))$ is the average cost of edges in E , and $\alpha \geq 0$ is a real parameter. In other words, an edge cover and is covered by another, when their incident nodes are close enough. Moreover, $D_\alpha(u, v)$ always include (u, v) independently of α .

Table 1 shows the data for the instance sets: the first column is the set name followed by the number of instances in the set in parentheses; $|V|$ and $|E|$ are the number of nodes and edges, respectively; the last three columns are the average size of covering functions D_α for $\alpha = 1$, $\alpha = 2$ and $\alpha = 3$ respectively.

Table 1: Data of instances.

Set	$ V $	$ E $	$ D_{\alpha=1} $	$ D_{\alpha=2} $	$ D_{\alpha=3} $
Alba(2)	90-102	144-160	2.8	9.8	18.4
P(24)	7-50	13-184	4.0	21.4	43.8
R(20)	20-50	37-203	3.8	28.0	53.8
D(36)	16-100	31-200	3.1	10.4	20.9
G(36)	16-100	24-180	1.0	7.6	18.4
UR500(12)	298-499	597-1526	3.6	15.6	35.2

Tables 2, 3 and 4 give the overall comparison of the branch-and-cut and GRP methods considering the covering functions D_α for $\alpha = 1$, $\alpha = 2$ and $\alpha = 3$ respectively. The first column is the instance set name and in parentheses the number of instances. For each method, OPT represents the number of instances solved to optimality; $Gap = 100 * (UB - LB)/LB$ is the average gap in percentage between the solution cost found by each method (UB) and the lower bound obtained by branch-and-cut (LB); and Time is the method runtime in seconds.

Table 2: Experiment results for covering function D_α with $\alpha = 1$.

Set	Branch-and-cut			GRH		
	OPT	Gap	Time	OPT	Gap	Time
Alba(2)	2	0.00	5.2	0	2.41	3.1
P(24)	24	0.00	18.4	12	1.05	0.6
R(20)	20	0.00	1.2	3	8.43	2.6
D(36)	36	0.00	1.9	9	3.18	1.9
G(36)	36	0.00	0.4	28	0.23	1.3
UR500(12)	1	1.19	1705.3	0	12.64	708.5
overall(130)	119	0.11	161.7	52	3.64	66.9

Table 3: Experiment results for covering function D_α with $\alpha = 2$.

Set	Branch-and-cut			GRH		
	OPT	Gap	Time	OPT	Gap	Time
Alba(2)	2	0.00	47.2	0	6.49	2.0
P(24)	24	0.00	3.0	21	1.68	0.5
R(20)	20	0.00	16.3	16	1.40	0.7
D(36)	29	0.85	398.2	6	5.83	1.1
G(36)	27	2.71	712.4	11	21.71	0.5
UR500(12)	0	323.30	1804.4	0	62.95	421.2
overall(130)	102	30.83	477.9	54	14.06	39.6

From Tables 2, 3 and 4 one can observe the following:

- Branch-and-cut method performed very well for all instances with $\alpha = 1$. In fact, 119 instances of 130 were solved to optimality and an average Gap of 0.11 was achieved in reasonable computational time.
- Branch-and-cut also performed well for the instances Alba, P, R, D and G with $\alpha = 2$. The computational time increased in comparison to $\alpha = 1$ for all instance sets.
- Branch-and-cut performed poorly for the UR500 instances with $\alpha = 2$ and $\alpha = 3$.
- Branch-and-cut performed competitively in comparison to GRH for all instances with $\alpha = 3$, except for UR500.

Table 4: Experiment results for covering function D_α with $\alpha = 3$.

Set	Branch-and-cut			GRH		
	OPT	Gap	Time	OPT	Gap	Time
Alba(2)	2	0.00	63.0	0	7.57	1.5
P(24)	24	0.00	1.1	24	0.00	0.1
R(20)	18	9.05	370.1	15	9.62	0.6
D(36)	33	0.24	369.1	12	4.21	0.8
G(36)	27	17.83	578.1	24	25.01	0.4
UR500(12)	0	2010.66	1801.1	0	227.70	181.6
overall(130)	104	191.99	486.7	75	30.71	17.2

- GRH performed much better than branch-and-cut for the UR500 instances with $\alpha = 2$ and $\alpha = 3$. In fact, the solutions obtained have *Gap* at least 5 times smaller and the processing time was at least 4 times faster.
- GRH is much faster than branch-and-cut for the majority of instance sets, for example the D and G instances with $\alpha = 2$ and $\alpha = 3$ was at least 360 times faster using GRH than branch-and-cut.
- GRH is faster for greater values of α . For example, the overall processing time for $\alpha = 1$ was 66.9s while for $\alpha = 3$ it was only 17.2s.
- GRH does not degrade its performance when considering instances with larger α values as the branch-and-cut does. For example, GRH obtained 52, 54 and 75 optimum solutions for $\alpha = 1$, $\alpha = 2$ and $\alpha = 3$ respectively.

One can conclude from these experiments that the proposed branch-and-cut method was especially effective for small and medium sized instances (up to 200 edges) and instances with narrower covering functions (small α). On the other hand, GRH seems to take advantage of wider covering functions in relation to processing time.

Tables 5, 6 and 7 show detailed experiment results for the UR500 instances, which are the largest instances. The first column is instance number; $|V|$ and $|E|$ are number of nodes and edges respectively; LB is the lower bound obtained by branch-and-cut method. For each method, UB is the solution cost and $Gap = 100 * (UB - LB) / LB$ is the average gap in percentage.

In Table 5 one can observe the very good performance of the branch-and-cut method for UR500 instances with $\alpha = 1$. The performance is not degraded when comparing the largest instances to the smaller ones. One optimal solution was proved.

In Table 6 and 7 the branch-and-cut display poor results in general for UR500 with $\alpha = 2$ and $\alpha = 3$. Branch-and-cut bad performance can be partly explained because it is an exponential-time algorithm in the worst case [Jünger et al., 2009]. Another possibly reason may be because the connectivity constraints (Section 4.1) perform worse for instances with wider covering functions.

On the other hand, GRH did not perform as bad as branch-and-cut for UR500 with $\alpha = 2$ and $\alpha = 3$. Therefore, it is observable some degree of complementarity in the proposed methods. In the case of wide covering functions, the branch-and-cut can be very sensitive and degrade its

Table 5: Experiment results detailed for UR500 instances and covering function D_α with $\alpha = 1$.

#	V	E	Branch-and-cut			GRH	
			LB	UB	Gap	UB	Gap
1	298	597	36513	37190	1.85	39298	7.63
2	458	812	30522	30832	1.02	34051	11.56
3	493	868	29823	29823	0.00	33978	13.93
4	343	862	44854	45603	1.67	50052	11.59
5	476	1104	40554	40873	0.79	45255	11.59
6	498	1112	37799	38213	1.10	42502	12.44
7	388	1135	49288	49854	1.15	56257	14.14
8	490	1305	46089	46471	0.83	51602	11.96
9	498	1310	44583	45154	1.28	50296	12.81
10	416	1403	55910	56754	1.51	64703	15.73
11	496	1513	56163	56702	0.96	63056	12.27
12	499	1526	51253	52368	2.18	59443	15.98
avg					1.19		12.64

Table 6: Experiment results detailed for UR500 instances and covering function D_α with $\alpha = 2$.

#	V	E	Branch-and-cut			GRH	
			LB	UB	Gap	UB	Gap
1	298	597	23925	25010	4.54	27262	13.95
2	458	812	16926	39161	131.37	20083	18.65
3	493	868	14967	15272	2.04	18091	20.87
4	343	862	21502	82704	284.63	28741	33.67
5	476	1104	14404	47580	230.32	22143	53.73
6	498	1112	12073	58790	386.95	18022	49.28
7	388	1135	18290	48617	165.81	26061	42.49
8	490	1305	8835	68768	678.36	17911	102.73
9	498	1310	10716	59074	451.27	17976	67.75
10	416	1403	12472	97698	683.34	22158	77.66
11	496	1513	6366	40454	535.47	16220	154.79
12	499	1526	8113	34521	325.50	17837	119.86
avg					323.30		62.95

performance; on the other hand, GRH compute faster its solutions and does not seem to degrade its performance because of wide covering functions.

7. Conclusion

In this paper we have proposed a new combinatorial optimization problem called covering Chinese postman problem (CCPP). The problem was modeled by integer programming. Two solution methods were proposed: an exact branch-and-cut method and a heuristic method called GRP. The proposed formulation included a exponential sized set of constraints; therefore, a separation algorithm was integrated to the branch-and-cut in order to add these constraints as they are needed in the optimization process; otherwise, the formulation would be restricted to very small instances.

Extensive computational experiments with six sets of benchmark instances have been performed. The results show the usefulness of proposed methods. For small and medium instances ($|E| \leq 200$) the branch-and-cut obtained very good results, including many optimal solutions. However for some large instances it was ineffective. On the other hand, the GRP method was able

Table 7: Experiment results detailed for UR500 instances and covering function D_α with $\alpha = 3$.

#	V	E	Branch-and-cut			GRH	
			LB	UB	Gap	UB	Gap
1	298	597	14933	50045	235.13	18226	22.05
2	458	812	6298	40205	538.38	12949	105.60
3	493	868	5824	38083	553.90	13063	124.30
4	343	862	9492	72590	664.75	15865	67.14
5	476	1104	6312	72296	1045.37	14129	123.84
6	498	1112	3137	64432	1953.94	10869	246.48
7	388	1135	3914	58532	1395.45	10220	161.11
8	490	1305	1721	54973	3094.25	9132	430.62
9	498	1310	2241	78888	3420.21	9716	333.56
10	416	1403	1499	62148	4045.96	6942	363.11
11	496	1513	1526	59138	3775.36	8249	440.56
12	499	1526	2039	71471	3405.20	8442	314.03
avg					2010.66		227.70

to obtain reasonable solutions in short computational time for all instances, including the largest ones, although for medium and small instances the branch-and-cut performed better.

In addition to the instance sizes, the covering function was also show to play a major role in instances difficulty. While instances with narrow covering functions tend to be well solved by the branch-and-cut, wide covering functions tend to be more difficult for the branch-and-cut method. The GRP was shown to perform better than branch-and-cut for large instances with wide covering functions.

Rather than proposing highly efficient methods, this paper intend to show the usefulness of the proposed approaches for the problem. Future works include improvements on the integer programming formulation by addition of valid inequalities. Another interesting research direction is the development of an efficient metaheuristic for the CCP, including local search procedures specially tailored for the problem.

References

- Altay, N. and Green III, W. G. (2006). Or/ms research in disaster operations management. *European journal of operational research*, 175(1):475–493.
- Aráoz, J., Fernández, E., and Meza, O. (2009). Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3):886–896.
- Belenguer, M. J. and Benavent, E. (2002). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30:705–728. ISSN 0305-0548.
- Christofides, N., Campos, V., Corberán, A., and Mota, E. (1981). An algorithm for the rural postman problem. Technical report.
- Corberán, A. and Sanchis, J. (1994). A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79(1):95–114.
- Corberán, A., Plana, I., and Sanchis, J. M. (2007). A branch & cut algorithm for the windy general routing problem and special cases. *Networks: An International Journal*, 49(4):245–257.

- Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition.
- Current, J. R. and Schilling, D. A. (1989). The covering salesman problem. *Transportation science*, 23(3):208–213.
- Dezső, B., Jüttner, A., and Kovács, P. (2011). Lemon—an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45.
- Drexl, M. (2014). On the generalized directed rural postman problem. *Journal of the Operational Research Society*, 65(8):1143–1154.
- Eiselt, H. A., Gendreau, M., and Laporte, G. (1995). Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242.
- Gomory, R. E. and Hu, T. C. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570.
- Hertz, A., Laporte, G., and Hugo, P. N. (1999). Improvement procedures for the undirected rural postman problem. *INFORMS Journal on computing*, 11(1):53–62.
- Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A. (2009). *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media.
- Minieka, E. (1979). The chinese postman problem for mixed networks. *Management Science*, 25(7):643–648.
- Ozbaygin, G., Yaman, H., and Karasan, O. E. (2016). Time constrained maximal covering salesman problem with weighted demands and partial coverage. *Computers & Operations Research*, 76: 226–237.
- Pearn, W. L. and Li, M. L. (1994). Algorithms for the windy postman problem. *Computers & operations research*, 21(6):641–651.
- Reina, D., Marin, S. T., Bessis, N., Barrero, F., and Asimakopoulou, E. (2013). An evolutionary computation approach for optimizing connectivity in disaster response scenarios. *Applied Soft Computing*, 13(2):833–845.
- Resende, M. G. and Ribeiro, C. C. (2014). Grasp: Greedy randomized adaptive search procedures. In *Search methodologies*, p. 287–312. Springer.
- Salari, M. and Naji-Azimi, Z. (2012). An integer programming-based local search for the covering salesman problem. *Computers & Operations Research*, 39(11):2594–2602.
- Welz, S. A. (1994). *Optimal solutions for the capacitated arc routing problem using integer programming*. PhD thesis, University of Cincinnati.