A Branch-and-Cut Algorithm for the Maximum Covering Cycle Problem

Eduardo Álvarez-Miranda*[1] and Markus Sinnl†[2]

[1] *Department of Industrial Engineering, Universidad de Talca, Curicó, Chile.*
[2] *ISOR, University of Vienna, Vienna, Austria.*

Abstract

In many applications in telecommunications and routing, we seek for cost-effective infrastructure or operating layouts so that many nodes (e.g., customers) of a support network (typically modeled by a graph) are covered by, or at least are easily reachable from, such layout. In this paper, we study the maximum covering cycle problem. In this recently introduced problem, we are given a non-complete graph, and the goal is to find a cycle, such that the number of nodes which are either on the cycle or are adjacent to the cycle is maximized. We design a branch-and-cut framework for solving the problem. The framework contains valid inequalities, lifted inequalities and a primal heuristic. In a computational study, we compare our framework to previous work available for this problem. The results reveal that our approach significantly outperforms the previous approach. In particular, all available instances from literature could be solved to optimality with our approach, most of them within a few seconds.

1. Introduction

Covering and domination problems in graphs have attracted the interest of researchers since at least the 1970s. In such type of problems we are concerned with finding a subset of nodes S of an input graph, such that a certain set of nodes are *covered* or *dominated*, i.e., either by belonging to S or by being adjacent to S. There exists many different versions of such problems, depending on topological constraints on S (e.g., it may need to form a tree or cycle), the nodes that are required to be covered, the objective function and, eventually, additional side constraints. While historically the focus has been on theoretical properties (see, e.g., [7, 17, 22]), recently, computational studies as well as the applications associated to these problems (ranging from telecommunication network design to facility location) have also been addressed (see, e.g., [1, 5, 19]).

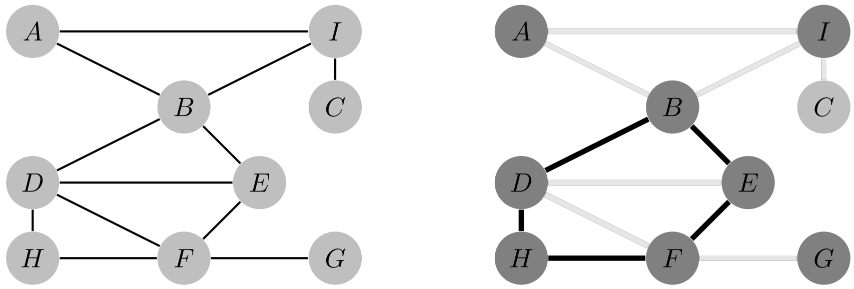
In this paper we look at the *maximum covering cycle problem* (MCCP), which was recently introduced by Grosso et al. [15]. The problems is defined as follows. Let $G = (V, E)$ be an undirected graph and $C \subseteq V$ be a cycle in G. The cycle C is said to *cover* a node $v \in V$, if either v is on the cycle C or it is adjacent to it. The goal of the MCCP is to find a cycle in G, which covers the maximum number of nodes. Let $\mathcal{C}$ be the set of all cycles of G and $f(V') = \{v \in V | v \in V' \vee \exists \{v, v'\} \in E : v' \in V'\}$. The problem, which was shown to be NP-hard in general graphs, can be formally stated as

$$\max |f(C)|$$
$$s.t., \quad C \in \mathcal{C}.$$

Note that the problem is trivial in complete graphs, as any single node covers all other nodes. It is also trivial if the graph is a star, as the center node of the star covers all nodes. Moreover, if the graph contains

*ealvarez@utalca.cl
†markus.sinnl@univie.ac.at

a Hamiltonian cycle, this Hamiltonian cycle is (one) optimal solution. Figure 1 shows an instance of the problem and a feasible solution.



(a) Instance of the MCCP (b) Solution of the MCCP

Figure 1: An exemplary instance and a feasible solution for the MCCP. The nodes covered in the solution are black, and the cycle constituting the solution is given by the black edges.

To solve the MCCP, the authors of [15] propose a cutting plane approach based on Integer Linear Programming (ILP), which we briefly sketch in the following. Let binary variable $u_i \in \{0,1\}$ be a binary variable so that $u_i == 1$ iff node $i \in V$ is *covered* by the cycle, and let $w_i \in \{0,1\}$ be a binary variable so that $w_i = 1$, iff node $i \in V$ is *on* the cycle. Let A be the set of arcs obtained by bi-directing the edges E, i.e., $A = \{(i,j),(j,i) : \{i,j\} \in E\}$. Thus, let $x_{ij} \in \{0,1\}$ be a binary variable, so that $x_{ij} = 1$ iff arc (i,j) is on the cycle. The following formulation for the MCCP follows classic modeling techniques from, e.g., the asymmetric traveling salesman problem (ATSP, see, e.g., [3]), and the generalized TSP (see, e.g., [11]);

$$\max \quad \sum_{i \in V} u_i \tag{1}$$

$$s.t. \qquad x_{ij} + x_{ji} \leq 1 \quad \forall \{i,j\} \in E \tag{2}$$

$$w_i + \sum_{j:\{i,j\} \in E} w_j \geq u_i \quad \forall i \in V \tag{3}$$

$$\sum_{(i,j) \in A} x_{ij} = w_i \quad \forall i \in V \tag{4}$$

$$\sum_{(j,i) \in A} x_{ji} = w_i \quad \forall i \in V \tag{5}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} + \sum_{j \in S} \sum_{j \in V \setminus S} x_{ij} \geq 2(w_k + w_l - 1)$$
$$\forall S \subset V, k \in S, l \in V \setminus S : 2 \leq |S| \leq |V| - 2 \tag{6}$$

$$u_i, w_i \in \{0,1\}, \ \forall i \in V \text{ and } x_{ij} \in \{0,1\}, \ \forall (i,j) \in A \tag{7}$$

Constraints (2) ensure that, for each edge, only one of the arc-variables x_{ij} and x_{ji} is taken. Constraints (3) make sure that either node i or at least one of its adjacent nodes is in the cycle, if i is chosen as covered. Constraints (4) and (5) ensure that every node on the cycle has exactly one ingoing and outgoing arc, respectively. Constraints (6) are subtour elimination constraints. these constraints are of exponential size. Finally, constraints (7) provide the variables nature. In the approach proposed in Grosso et al. [15], the authors remove the subtour or cycle elimination constraints (6) to obtain a compact formulation. Integer solutions to this relaxed formulation give a set of disjoint cycles. Hence, their algorithm works by iteratively solving the relaxed formulation (to integer optimality). The cycle giving the largest coverage is stored as incumbent solution, and the constraints forbidding the cycles found in the previous iterations are added. The algorithm terminates when the objective of the relaxed formulation (with the added cycle elimination constraints) is larger then the objective of the incumbent. Note that this proves optimality of the incumbent and that the algorithm is finite. The authors enhance their approach by heuristically creating additional cycles, for which the corresponding cycle elimination constraints are also added.

Contribution and Paper Outline In this paper, we develop a solution framework for the MCCP based on ILP. The framework is based on an exponential-sized ILP formulation, which is solved by means of a branch-and-cut scheme. A computational study on instances from the literature and additional instances is carried out in order to assess the efficiency of our approach. The study shows that our algorithm outperforms the approach by Grosso et al. [15]. Moreover, most of the instances can be solved within a few seconds.

In the remainder of this section, we discuss related work. In Section 2 we present our ILP model along with additional valid inequalities. Section 3 contains the description of our algorithmic framework, including separation algorithms and a primal heuristic. The computational results are discussed in Section 4. Finally, concluding remarks are presented in Section 5.

Related Work As already mentioned in the introduction, connected covering and domination problems have been studied for a long time. Depending on the application, authors seek for different covering (resp. dominating) topologies and coverage (resp. domination) protocols.

Complementary, there are of course many problems in literature which are concerned with finding cycles in a graph, being the *travelling salesman problem* (TSP) the most famous of them (see, e.g., [10, 18]). In the TSP, we look for a minimum cost Hamiltonian cycle through all the nodes in the graph. A variant of the TSP, with covering aspects, is the *covering salesman problem* (CSP), in which the goal is to find a minimum cost cycle (a tour), such that every node in the graph is within a certain distance from the cycle. The problem has been introduced by Current and Schilling [8], where a heuristic is presented. An approximation algorithm for the geometric version of the CSP is presented in Arkin and Hassin [2], and multi-objective variants of the problem are considered in Current and Schilling [9]. Furthermore, generalized versions of the CSP have been proposed, for instance, in [14], [24] and [25]. In Gendreau et al. [13], the *covering tour problem* (CTP) is presented. In the CTP, we are concerned with finding a minimum cost tour, which must go through a given subset of nodes, while the remaining nodes may or may not be on the tour . A bi-objective variant of the CTP is studied by Jozefowiez et al. [20].

We note that TSP-like problems are usually defined on a complete graph. The MCCP is of course trivial on such graphs, as any single node already covers all the nodes.

2. ILP-Model and Valid Inequalities

Our approach is based on a slightly different ILP formulation compared to the one proposed in [15]. In our formulation, we do not bi-direct the edges and, instead, work on the original graph. Let redefine binary variables $x_{ij} \in \{0,1\}$, so that $x_{ij} = 1$ iff edge $\{i,j\} \in E$ is in the cycle. Additionally, let $y_i \in \{0,1\}$ be binary variables such that $y_i = 1$, if node i is on the cycle; and let $z_i \in \{0,1\}$ be binary variables so that $z_i = 1$, if node i is *not* on the cycle, but *covered* by the cycle (i.e., it is covered by being adjacent to a node on the cycle). Finally, for $S \subset V$, let $\delta(S) = \{\{i,j\} \in E : i \in S, j \in V \setminus S\}$. Considering all these elements, our formulation reads as follows:

$$\max \sum_{i \in V} (y_i + z_i) \tag{OBJ}$$

$$\text{s.t.} \quad y_i + z_i \leq 1, \ \forall i \in V \tag{YZ}$$

$$\sum_{j:\{i,j\} \in E} y_j \geq z_i, \ \forall i \in V \tag{COV}$$

$$\sum_{\{i,j\} \in E} x_{ij} = 2y_i, \ \forall i \in V \tag{DEG}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_k + y_l - 1)$$

$$\forall S \subset V, k \in S, l \in V \setminus S : 2 \leq |S| \leq |V| - 2. \tag{SEC}$$

The objective function (OBJ) is the sum of nodes directly on the cycle, and the additional nodes covered by the cycle. Constraints (YZ) ensure, that a node cannot be both on the cycle, and covered by adjacency to the cycle. Constraints (COV) ensure that if a node i is selected to be covered by adjacency to the cycle, at least one adjacent node of i is in the cycle. Constraints (DEG) ensure that every node on the cycle has two adjacent edges. The subtour-elimination constraints are encoded by (SEC); since there is an exponential number of them, we separate them on-the-fly. The separation procedure is described in Section 3.1.

The following so-called *logic inequalities* (see, e.g., [11, 12]) are valid for our formulation;

$$x_{ij} \leq y_i, \quad \forall \{i,j\} \in E, i \in V. \tag{LOG}$$

Moreover, the subtour elimination constraints (SEC) can be lifted in some cases, as detailed by the following proposition.

Proposition 1. *Let $S \subset V, k \in S, l \in V \setminus S : 2 \leq |S| \leq |V| - 2$. Let $K = \{v \in V | \{k,v\} \in E\}$ and suppose, $K \subseteq S$. Then the following inequalities are valid*

$$\sum_{e \in \delta(S)} x_e \geq 2(y_k + z_k + y_l - 1). \tag{L-SEC}$$

Proof. Due to constraints (YZ) only one of y_k and z_k can be one in a feasible solution. If y_k is one, the inequalities reduce to inequalities (SEC). If instead $z_k = 1$, for at least one of the nodes in K, say m, we must have $y_m = 1$, due to (COV). As $m \in S$, any feasible solution must fulfill (SEC) defined by m, k and S, from which follows the validity of (L-SEC). □ □

Let $L = \{v \in V | \{l,v\} \in E\}$. If $L \subseteq V \setminus S$, using the same arguments, (L-SEC) can be further lifted by additionally adding z_l to the term in the parenthesis on right-hand-side. Moreover, if a feasible solution to the problem is available, an additional lifting is possible as shown in the following proposition.

Proposition 2. *Let $S \subset V, k \in S, l \in V \setminus S : 2 \leq |S| \leq |V| - 2$. Let $\bar{Z}$ be the objective value of a feasible solution. Let $K_2 = \{v \in V | v \in V \setminus S \cup v : \exists \{v, v'\} \in E : v' \in V \setminus S\}$, i.e., the set $V \setminus S$ and all nodes adjacent to it. If $|\bar{Z}| > |K_2|$, the inequality*

$$\sum_{e \in \delta(S)} x_e \geq 2y_l, \tag{L2-SEC}$$

is valid for the problem.

Proof. Due to $|\bar{Z}| > |K_2|$, the optimal solution cannot be a cycle on the nodes in $V \setminus S$ (plus nodes adjacent to this cycle). Thus, at least one node in S must be in an optimal solution. □ □

Likewise, let $L_2 = \{v \in V | v \in S \cup v : \exists \{v, v'\} \in E : v' \in S\}$, i.e., the set S and all nodes adjacent to it. If $|\bar{Z}| > |L_2|$, a similar lifting is possible. If both $|\bar{Z}| > |K_2|$ and $|\bar{Z}| > |L_2|$, the right-hand-side in (L2-SEC) can be lifted to two; this means that a solution with objective value of at least $\bar{Z}$ cannot be a cycle contained only in S or $V \setminus S$, but must contain nodes from both S and $V \setminus S$. Finally, if it holds that $|\bar{Z}| > |K_2|$ and $|\bar{Z}| > |L_2|$, then inequalities (L2-SEC) can also be lifted by using the arguments of (L-SEC) for node l; in this case, the term $2z_l$ can be added to the right-hand-side.

3. Algorithmic Framework

In this section, we give implementation details of our algorithmic framework. Namely, of the separation procedure for inequalities (SEC) (and its lifted versions), and also the primal heuristic. In contrast to TSP-like problems, where inequalities (LOG) are usually separated by enumeration, we add them directly at the initialization, as the instances we are dealing with are sparse (while TSP-like problems normally consider complete graphs).

3.1 Separation Algorithms

Let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ be the solution of the LP-relaxation at a branch-and-bound node. Depending on whether this solution is fractional or integer, different separation strategies are employed.

Separation of (SEC) **for integer solutions.** If the solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ is integer, it forms a set of disjoint cycles, say $C^1, \dots C^n$. For each cycle C^i, we add an inequality (SEC) with $S = C^i$. As node k, we randomly chose a node in C^i; as node l, we randomly chose a node in C^{i+1} (C^0, if $i = n$).

Separation of (SEC) **for fractional solutions.** Inequalities (SEC) can be separated in polynomial time, $O(|V|^4)$, by maximum flow-computations (see, e.g., [11]). As this turned out to be too time-consuming in preliminary computations, we used a heuristic separation instead. In particular, we used the heuristic from [13]: Construct a maximum spanning tree on G (with edge weights given by the $\mathbf{x}^*$ values) using a greedy algorithm (in our implementation, we used Kruskal's algorithm [23]). Whenever an edge e gets added to the partial tree during the algorithm, we take S, which is the set where e gets added to, as candidate for a violated inequality (SEC). As node k, we randomly chose a node in S among all nodes with maximum y^* value in S, and for l, we randomly chose a node in $V \setminus S$ among all nodes with maximum y^* value in $V \setminus S$.

Separation of (L-SEC) **and** (L2-SEC)**.** We do not separate (L-SEC) and (L2-SEC) explicitly, but whenever we detect a violated inequality (SEC), we check if a lifting is possible. This can be simply done by checking, if any $v \in S$ or $V \setminus S$ fulfills the condition for (L-SEC), and if S or $V \setminus S$ fulfills the condition for (L2-SEC).

3.2 Primal Heuristic

In order to find high-quality solutions during the branch-and-cut, we implemented a primal heuristic, which is driven by the values of the LP-relaxation in the branch-and-bound nodes. The heuristic works on the support graph G^* induced by the LP-solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$, i.e., the graph induced by $E^* = \{\{i, j\} \in E | x^*_{ij} > 0\}$. Using $\mathbf{x}^*$ as weights, we compute a maximum spanning tree on G^* using Kruskal's algorithm. Whenever adding an edge during the course of the algorithm, would induce a cycle C, we check if C gives an improved primal solution; if yes, we take C as new incumbent.

In addition to the above primal heuristic, whenever we encounter an infeasible integer solution (which forms a disjoint set of cycles) during the branch-and-bound, we check if any cycle in this solution gives an improved primal solution (and we take it as incumbent).

4. Computational Results

We implemented our framework in C++ and used CPLEX 12.7 for the branch-and-cut. All CPLEX parameters were left at their default values. The experiments were done on a Xeon CPU with 2.5 GHz using a single-thread. For each run, we used a timelimit of 600 seconds and a memorylimit of 3GB. The study in [15] used the same timelimit, but unfortunately does not mention the specifications of the computer.

As benchmark instances, we used a subset of the instances from [15] and also considered additional instances. We were not able to consider all instances from [15], as not all were available online. Details of the instance sets are given next:

- `Coloring`: These are graph coloring instances from ORLIB [4] at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/colourinfo.html`. In [15], the authors state that they used 57 instances from there. However, directly on ORLIB, we only found 30 instances (`gcol1` to `gcol30`). Additionally, we included the 79 graph coloring instances available at `http://mat.gsia.cmu.edu/COLOR/instances.html` to this set (this webpage was linked in the graph coloring section at ORLIB).

- `Benchmark/Random-HC-DLV`: These are instances containing a Hamiltonian cycle from `http://wwwinfo.deis.unical.it/npdatalog/experiments/hamiltoniancycle.htm`. Since they contain a Hamiltonian cycle, the value of the optimal solution is $|V|$ for all of these instances, and the Hamiltonian cycle is an optimal solution (there can also be other optimal solutions). These instances were used in [15].

- `Structured-3col-DLV`: These are graph 3-coloring instances from `http://wwwinfo.deis.unical.it/npdatalog/experiments/3-coloring.htm`. As well as in [15], we consider three instances from there, the largest one being comprised by 900 nodes.

- `Scale-Free`: In [15], the authors created instances based on scale-free graphs (see, e.g., [6]) with up to 1000 nodes. According to their computational experience, these instances turned out to be the most difficult ones. As this set of instances was not available online, we created a set of 20 of such instances (ten with 1000 nodes and ten with 5000 nodes) using the scale-free graph generator available in NetworkX [16]. The instances are made available online at `https://msinnl.github.io/`.

- `ES250/500FST`: As the results below will reveal, our approach can solve all instances from, resp., based on the ones in [15] within a few seconds. Thus, in order to push our approach to the limit, we looked at many different graph instances sets available online. In preliminary tests, the set `ES250/500FST` of SteinLIB [21], a library of Steiner tree instances, available at `http://steinlib.zib.de/showset.php?ES250FST` and `http://steinlib.zib.de/showset.php?ES500FST`, proved to be the most challenging of sets containing graphs with a reasonable size (i.e., large enough for yielding difficult ILP problems, but for which solving the linear relaxation is still not burdensome). This set contains 30 instances.

4.1 Effects of the Framework Ingredients

To study the effects of the proposed enhancements in our framework, we tested the following settings:

- `b`: Basic setting, where we do not use the primal heuristic and only separate (SEC) for integer solutions.

- `bh`: `b`, but with the primal heuristic.

- `bhf`: Using the primal heuristic and separation of (SEC) also for fractional solution, but no lifting of (SEC).

- `bhfl`: `bhf`, where we also lift inequalities (SEC).

In Figure 2, we show a performance profile plot of the runtime to optimality for all instances and all settings. Likewise, in Figure 3, we show the performance profile of the optimality gap, which is calculated as $100 \cdot (z^{DB} - z^*)/z^*$, where z^{DB} is the dual bound and z^* is the best solution found.

From Figure 2 we see that all settings manage to solve about 75% of the instances within ten seconds. In general, for most of the instances, there is not much difference in the runtime to optimality between the different settings, i.e., our branch-and-cut approach is already very effective in its most basic version. However, setting `bhfl` manages to solve a few more instances to optimality within the timelimit. Regarding the instances which cannot be solved within the given timelimit (they are all of the set `ES250/500FST`, as we will describe in next section), we can see, from Figure 3, more pronounced differences between the settings. Every additional ingredient added in our framework improves the performance. Therefore, we conclude that setting `bhfl`, which contains all enhancements, is the most effective one for the considered problem. Therefore, all the results reported in the remainder of the section are obtained with this setting.

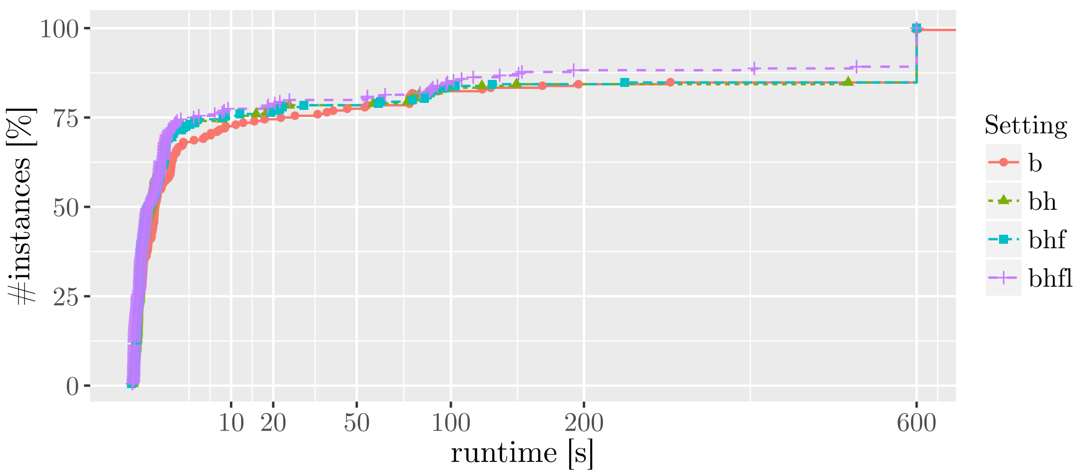


Figure 2: Runtime for different settings

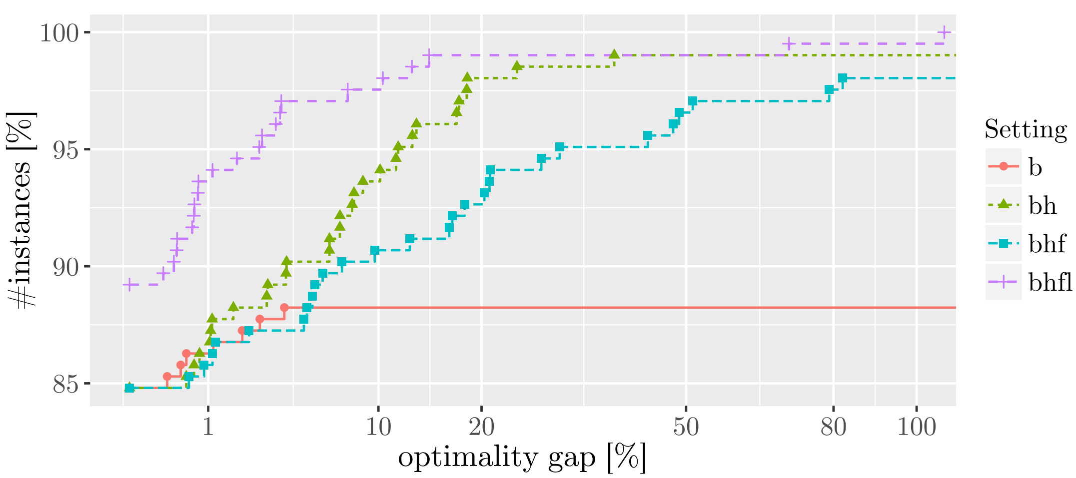


Figure 3: Optimality gap for different settings

4.2 Detailed Results

In Table 1 we give an overview of the results attained for each instance set. Column $t[s]$ gives the runtime in seconds, $g[\%]$ gives the optimality gap, #BBn gives the number of branch-and-bound nodes, #(SEC) gives the number of separated subtour elimination constraints, while columns #(L-SEC) and #(L2-SEC) report how often each of the corresponding lifting was successful. Note that for one subtour elimination constraint, both lifting strategies could be applied (both (L-SEC) and (L2-SEC) can be applied twice; for S and $V \setminus S$). The entries associated to each row the table are averages over the corresponding whole instance set. From the results, we see that all instances, except set `ES250/500FST`, could be solved to optimality. In particular, and in contrast to [15], all three instances from `Structured-3col-DLV` could be solved to optimality. Moreover, all instances from `Coloring` could also be solved to optimality, while [15] did only manage to solve 53 out of 57 of this class (recall that we took a superset of the instances from [15]). The runtime and also the number of branch-and-bound nodes for all instance classes except `ES250/500FST` is very small. All instances from sets `Benchmark/Random-HC-DLV` and `Scale-Free` were solved in the rootnode.

Table 1: Overview of results by instance class. The table contains the mean values over all instances of a class.

set	t [s]	g [%]	#BBn	#(SEC)	#(L-SEC)	#(L2-SEC)
`Coloring`	3.82	0.00	2.60	77.09	57.22	51.85
`Benchmark/Random-HC-DLV`	0.02	0.00	0.00	1.84	0.37	0.84
`Structured-3col-DLV`	18.94	0.00	15.67	1869.00	2166.67	1067.67
`Scale-Free`	54.02	0.00	0.00	37.60	49.05	8.25
`ES250/500FST`	496.82	8.23	4207.00	19763.77	15104.67	23624.10

In Table 2 we give a detailed overview on the results on the instances from set `ES250/500FST`, which is the only class where our algorithm did not find the optimal solution for all instances within the timelimit (these cases are indicated by **TL** in column $t[s]$). We manage to solve eight out of 30 instances of this set for optimality. For all, except five instances, the gap is under ten percent. In particular, for most of the instances, the gap is under one percent.

Table 2: Detailed results for instance set `ES250/500FST`.

name	Nodes	Edges	t [s]	UB	LB	g [%]	#BBn	#(SEC)	#(L-SEC)	#(L2-SEC)
es250fst01	623	876	**TL**	544	543	0.18	16194	7270	4342	9800
es250fst02	542	719	**TL**	476	473	0.63	16315	16784	6974	25968
es250fst03	543	727	**TL**	310	182	70.19	4940	25464	38710	10412
es250fst04	604	842	81.66	577	577	0.00	1000	12598	7177	17575
es250fst05	596	832	**TL**	526	522	0.77	18462	7650	4463	10473
es250fst06	596	824	378.03	493	493	0.00	11169	13993	7598	19717
es250fst07	585	799	**TL**	550	548	0.36	6214	22924	8123	36952
es250fst08	657	947	**TL**	637	630	1.11	4684	23771	12186	34622
es250fst09	570	770	**TL**	534	530	0.75	22173	10574	3626	17135
es250fst10	662	951	7.03	558	558	0.00	37	3917	4847	2906
es250fst11	661	952	**TL**	522	252	107.14	16	25337	41379	8050
es250fst12	619	872	**TL**	560	558	0.36	6607	16366	6752	25531
es250fst13	684	993	63.24	589	589	0.00	1060	7501	4729	9886
es250fst14	710	1046	**TL**	633	631	0.32	3260	20749	13040	27678
es250fst15	713	1053	24.75	640	640	0.00	321	6016	4912	6778
es500fst01	1250	1763	**TL**	1144	1103	3.72	1682	24772	21644	26727
es500fst02	1408	2056	**TL**	1347	1302	3.46	902	21837	17413	25205
es500fst03	1337	1933	**TL**	1264	1104	14.49	667	32613	35753	28256
es500fst04	1296	1879	**TL**	1247	1158	7.69	471	35571	21095	49193
es500fst05	1172	1627	**TL**	1088	1058	2.84	1388	30125	18743	40055
es500fst06	1335	1932	**TL**	1286	1252	2.72	261	24732	20286	27897
es500fst07	1214	1700	**TL**	1133	1093	3.66	1730	22936	16463	28476
es500fst08	1349	1972	**TL**	1313	1289	1.86	909	25311	18441	31526
es500fst09	1294	1853	**TL**	1175	1041	12.87	420	34481	22792	44903
es500fst10	1203	1679	97.04	1050	1050	0.00	644	9179	6670	11155
es500fst11	1274	1808	**TL**	1187	1179	0.68	1360	29718	16295	42197
es500fst12	1322	1918	**TL**	1289	1168	10.36	599	21908	24017	19149
es500fst13	1273	1814	**TL**	1210	1202	0.67	2206	24300	15851	31814
es500fst14	1477	2204	132.70	1436	1436	0.00	0	10948	10729	10446
es500fst15	1334	1927	511.88	1263	1263	0.00	519	23568	18090	28241

5. Conclusion

In many applications, such as telecommunications and routing, we are concerned in finding layouts so that many nodes (e.g., customers) of the underlying graph are covered. In this paper, we study the maximum covering cycle problem (MCCP), which has been recently introduced by [15]. In the MCCP we are given a (non-complete) graph, and the goal is to find a cycle such that the number of nodes which are either on the cycle or adjacent to this cycle is maximized. We design and implement a branch-and-cut framework for the problem. The framework contains valid inequalities, lifted inequalities and a primal heuristic. In a computational study, we compare our framework to the approach by [15]. The results reveal that our approach significantly outperforms the previous approach. In particular, all available instances from literature could be solved to optimality with our approach.

Regarding further work, the formulation can be easily extended to accommodate for a weighted coverage. Likewise, the coverage protocol can be extended to more general concepts. However, the primal heuristic may need non-trivial adaptations to satisfactory work in such cases, and additional valid inequalities could potentially be derived. It could also be interesting to investigate, if there are certain graphs, where the proposed formulation gives a complete description. Developing preprocessing tests to reduce the instance size could be useful. Furthermore, in a real-life setting, is likely that both the set of nodes and links are subject to uncertainty; therefore, the study of a stochastic or robust version of the problem could be an worthwhile topic for research. For large-scale instances, solving the Integer Linear Programming formulation can become a bottleneck, thus the use of Lagrangian relaxation instead of Linear Programming may prove fruitful to quickly find reasonable dual bounds. The design of (meta)-heuristic approaches to tackle even larger instances could also be interesting for further work.

Acknowledgements

E.A.-M. acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the FONDECYT Grant N.11140060 and through the Complex Engineering Systems Institute (ICM-FIC:P-05-004-F, CONICYT:FB0816). The research of M.S. was supported by the Austrian Research Fund (FWF, Project P 26755-N19).

References

[1] A. Aazami. Domination in graphs with bounded propagation: algorithms, formulations and hardness results. *Journal of combinatorial optimization*, 19(4):429–456, 2010.

[2] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

[3] E. Balas. The asymmetric assignment problem and some new facets of the traveling salesman polytope on a directed graph. *SIAM Journal on Discrete Mathematics*, 2(4):425–451, 1989.

[4] J. E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, pages 1069–1072, 1990.

[5] A. Bley, I. Ljubić, and O. Maurer. A node-based ilp formulation for the node-weighted dominating steiner problem. *Networks*, 69(1):33–51, 2017.

[6] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete Algorithms*, pages 132–139. Society for Industrial and Applied Mathematics, 2003.

[7] C. Colbourn and L. Stewart. Permutation graphs: Connected domination and Steiner trees. In S. Hedetniemi, editor, *Topics on Domination*, volume 48 of *Annals of Discrete Mathematics*, pages 179–189. 1991.

[8] J. R. Current and D. A. Schilling. The covering salesman problem. *Transportation Science*, 23(3): 208–213, 08 1989.

[9] J. R. Current and D. A. Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114–126, 1994.

[10] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[11] M. Fischetti, J. Salazar-González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[12] M. Fischetti, J. Salazar-González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1999.

[13] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45(4): 568–576, 1997.

[14] B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari, and P. Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

[15] A. Grosso, F. Salassa, and W. Vancroonenburg. Searching for a cycle with maximum coverage in undirected graphs. *Optimization Letters*, 10(7):1493–1504, 2016.

[16] A. Hagberg, P. Swart, and D. Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), 2008.

[17] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Pure and Applied Mathematics. CRC Press, 1st edition, 1998.

[18] K. L. Hoffman, M. Padberg, and G. Rinaldi. Traveling salesman problem. In *Encyclopedia of operations research and management science*, pages 1573–1578. Springer, 2013.

[19] I. Jeong. An optimal approach for a set covering version of the refueling-station location problem and its application to a diffusion model. *International Journal of Sustainable Transportation*, 11(2):86–97, 2017.

[20] N. Jozefowiez, F. Semet, and E. Talbi. The bi-objective covering tour problem. *Computers & Operations research*, 34(7):1929–1942, 2007.

[21] T. Koch, A. Martin, and S. Voß. SteinLib: An updated library on Steiner tree problems in graphs. *Steiner trees in industry*, 11:285–326, 2001.

[22] J. Kratochv, A. Proskurowski, and J. Telle. Complexity of graph covering problems. *Nordic Journal of Computing*, 5:173–195, 1998.

[23] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[24] G. Ozbaygin, H. Yaman, and O. Karasan. Time constrained maximal covering salesman problem with weighted demands and partial coverage. *Computers & Operations Research*, 76:226–237, 2016.

[25] M. Shaelaie, M. Salari, and Z. Naji-Azimi. The generalized covering traveling salesman problem. *Applied Soft Computing*, 24:867–878, 2014.