RAIRO-Oper. Res. 57 (2023) 1149–1166
https://doi.org/10.1051/ro/2023055

RAIRO Operations Research
www.rairo-ro.org

BRANCH-AND-CUT ALGORITHMS FOR THE COVERING SALESMAN PROBLEM

Lucas Porto Maziero*, Fábio Luiz Usberti and Celso Cavellucci

Abstract. The Covering Salesman Problem (CSP) is a generalization of the Traveling Salesman Problem in which the tour is not required to visit all vertices, as long as all vertices are covered by the tour. The objective of CSP is to find a minimum length Hamiltonian cycle over a subset of vertices that covers an undirected graph. In this paper, valid inequalities from the generalized traveling salesman problem are applied to the CSP in addition to new valid inequalities that explore distinct aspects of the problem. A branch-and-cut framework assembles exact and heuristic separation routines for integer and fractional CSP solutions. Computational experiments show that the proposed framework outperformed methodologies from literature with respect to optimality gaps. Moreover, optimal solutions were proven for several previously unsolved instances.

Mathematics Subject Classification. 68R05, 90B06, 90C10, 90C57.

Received July 29, 2022. Accepted April 10, 2023.

1. Introduction

Consider a set of sites scattered in the plane that must be covered by a single-vehicle tour. Knowing that each site covers some of its neighbors, what is the minimum length of an enclosed vehicle tour in which all sites are covered? This question is addressed by the Covering Salesman Problem (CSP), proposed by Current and Schilling [5] in 1989. More formally, given an undirected graph, the CSP objective is to find the shortest Hamiltonian cycle on a subset of vertices that covers the graph. The special case where each vertex covers strictly itself is the Travelling Salesman Problem (TSP) [2], which follows that CSP is also NP-hard.

Since its proposal, the CSP has attracted the attention of researchers due to its complexity and numerous applications. These applications arise in scenarios where it is unrealistic to visit all locations, *e.g.*, rural health services, areas affected by natural disasters, or planning mobile service units [5].

Several variants of CSP were investigated. Current *et al.* [7] studied the Shortest Covering Path Problem (SCPP). The goal is to find a minimum cost s–t path in a network that covers all vertices. The authors proposed two methods to solve the SCPP: a Lagrangian relaxation and a branch-and-bound algorithm that makes use of the obtained dual bounds.

Current and Schilling [6] introduced two bi-criterion routing problems: the Median Tour Problem (MTP) and the Maximal Covering Tour Problem (MCTP). Assuming a network with n vertices and a value p ($p \leqslant n$), the

Keywords. Covering salesman problem, integer linear programming, branch-and-cut algorithm.

Institute of Computing, University of Campinas, Av. Albert Einstein 1251, 13083-852 Campinas, SP, Brazil.
*Corresponding author: lucasporto1992@gmail.com

criteria for both problems are (i) to find a minimum length tour that visits exactly p of the n vertices and (ii) to maximize the accessibility of the vertices that are not in the tour. The problems differ in the way the accessibility of the second criterion is evaluated. In MTP, the second criterion is to minimize the sum of distances from each unvisited vertex to its closest vertex in the tour. In MCTP, the second criterion is to minimize the number of uncovered vertices. The authors proposed mathematical formulations and heuristics to solve both MTP and MCTP. Their methodologies were tested on a real-life scenario requiring the optimal location and sequence of stops for overnight mail service.

Another variant of CSP, studied by Gendreau *et al.* [13], is the Covering Tour Problem (CTP). Let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the set of vertices and E is the set of edges. Vertex v_0 is the depot, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited ($v_0 \in T$), and W is the set of vertices that must be covered but cannot be visited. The goal of the CTP is to determine a minimum length tour that visits a subset of vertices $S \subseteq V$ such that $T \subseteq S$ and each vertex of W is covered by some vertex in S. The authors proposed heuristics and a branch-and-cut algorithm to solve the CTP.

Golden *et al.* [14] proposed a generalized version of the CSP called the Generalized Covering Salesman Problem (GCSP). Given an undirected graph $G = (V, E)$, each vertex $i \in V$ has a covering demand k_i, meaning vertex i has to be covered at least k_i times. In addition, there is a fixed cost F_i that incurs when the tour visits vertex i. The objective of the GCSP is to minimize the solution cost which is given by the sum of the tour length and the costs of the visited vertices. The authors developed local searches that explore exchange, removal, and insertion of tour vertices to escape from local optimum.

Another similar problem to the CSP is the Generalized Traveling Salesman Problem (GTSP). In GTSP, the vertices are partitioned into disjoint subsets, called clusters, and the goal is to determine the minimum length tour that visits exactly one vertex from each cluster. The GTSP is a special case of the CSP, where each cluster can be modeled as a subset of vertices that mutually cover themselves. Fischetti *et al.* [12] propose a branch-and-cut algorithm based on exact and heuristic separation routines for some families of valid inequalities for the GTSP. These inequalities are translated for the CSP in Section 3.

Zhang and Xu [26] proposed the online CSP, where the vehicle will face up to k blocked edges not known *a priori* during its tour traversal. The objective is to find a minimum length tour that covers all vertices while bypassing the blocked edges. The authors presented a $(k+\alpha)$-competitive algorithm, where $\alpha = \frac{1}{2} + \frac{(4k+2)r}{OPT} + 2\upsilon\rho$, υ is the approximation ratio for the Steiner Tree Problem, ρ is the maximum number of vertices that can cover an arbitrary vertex and r is the radius which defines the covering neighbourhood of each vertex.

Many works in literature have given attention to the geometric CSP, also known as the Close Enough Traveling Salesman Problem (CETSP). In this version, each vertex has its neighborhood defined as a compact region of the plane. The goal is to find a minimum length tour that starts from a depot and intercepts all neighborhood sets, thus covering all its corresponding vertices. Approximation algorithms, heuristics and methodologies based on ILP were developed for this version [3, 4, 10, 11, 15, 21].

Table 1 emphasizes the main differences between CSP and its counterparts. In CTP, among the vertices that can be visited, for some of them the visitation is mandatory. As for the vertices that must be covered, in CTP these vertices cannot be visited. In GTSP, the vertices are clustered into disjoint neighborhoods, meaning each vertex covers exactly the vertices in the cluster it belongs. The vertices in GCSP may require multiple coverings and each visitation incurs into a fixed cost. Finally, in CETSP the vertices are covered by a compact region on the plane instead of being covered by a subset of vertices. All of these problems, despite sharing the idea of joining vehicle routing with set covering, contain important distinctions with respect to CSP. This explains why this problem still requires customized exact and heuristic methodologies.

Some solution methodologies were proposed in the literature for the CSP. Current and Schilling [5], for example, developed a two-step heuristic to solve the CSP: the first step solves a set cover problem; the second step solves the TSP on the vertices determined by the first step. More than two decades later Salari and Naju-Azimi [19] revisited the problem by proposing a heuristic for the CSP embedded within an integer linear programming (ILP) framework. First, they employ constructive heuristics to find good initial solutions and then the tour vertices are rearranged by the use of ILP techniques in an attempt to reduce its length. Salari

TABLE 1. Summary of the main differences between CSP, CTP, GTSP, GCSP and CETSP.

	Required/Forbidden visitations	Disjoint neighborhoods	Multiple coverings	Geometric neighborhood
CSP	✗	✗	✗	✗
CTP	✓	✗	✗	✗
GTSP	✗	✓	✗	✗
GCSP	✗	✗	✓	✗
CETSP	✗	✗	✗	✓

et al. [20] gave a polynomial-size formulation and a hybrid heuristic for the CSP, which combines ant colony optimization and dynamic programming. The formulation of Salari *et al.*, to the best of our knowledge, composes the state-of-the-art exact methodology for the CSP.

Venkatesh *et al.* [24] proposed a multi-start iterated local search (MS-ILS) algorithm for the CSP with a variable degree of perturbation. Computational results show that the proposed approach is competitive with other state-of-the-art heuristic approaches. Zang *et al.* [25] reformulated the CSP as a bilevel CSP with a leader and a follower sub-problem and proposed two parallel variable neighborhood search (PVNS) heuristics, namely, synchronous "master–slave" PVNS and asynchronous cooperative PVNS. Computational results show that the PVNS has improved previously best known solutions. Pandiri *et al.* [17] developed two hybrid metaheuristic approaches for the CSP. The first is based on the artificial bee colony algorithm and the second is based on the genetic algorithm. Both approaches were competitive with the state-of-the-art heuristics. Lu *et al.* [16] presented a hybrid evolutionary algorithm (HEA) that assembles a crossover operator with solution reconstruction, a destroy-and-repair mutation operator and a two-phase tabu search. The HEA also uses Lin-Kernighan local search on multiple stages. Computational results show that for 21 out of the 27 large instances, the HEA has improved previously best known solutions, while for small and medium instances the HEA has achieved previously best known solutions.

Recent applications of the CSP concern the routing of drones, which involves additional operational constraints. For example, Vásquez *et al.* [23] studied the Travelling Salesman Problem with Drone (TSP-D). Given a complete digraph, each node must be visited either by a vehicle route starting and ending at a depot or by a drone executing dispatches to customers from the vehicle during the route. When a drone visits a customer, it must fly back to meet the vehicle at a scheduled location on its route. The objective is to minimize travel time. The authors proposed a mixed integer programming formulation and valid inequalities for the TSP-D, solving it by Bender's decomposition using a two-stage approach: first selecting a subset of customers to be visited by the vehicle and then defining where the drone will be dispatched for the remaining customers. Computational experiments validated the performance of the algorithm using a benchmark of instances randomly generated.

A generalization of the TSP-D that considers more than one drone is the Travelling Salesman Problem with Multiple Drones (TSP-MD). Tiniç *et al.* [22] proposed a flow-based and a cut-based mixed integer linear programming formulations for TSP-MD. Also, branch-and-cut algorithms were developed for relaxations of the proposed formulations. The computational experiments showed that the branch-and-cut approaches outperformed the flow-based formulation. A sensitivity analysis was also made with various problem parameters, exploring scenarios such as vehicle operating cost, speed, and drone endurance.

The Flying Sidekick Travelling Salesman Problem (FSTSP) is another drone routing problem similar to the TSP-D. In FSTSP, there are some nodes that can be visited only by the vehicle, and each drone flight is limited by battery duration. Dell'Amico *et al.* [8] proposed a branch-and-bound exact algorithm and a heuristic, testing them on instances from the literature. The results showed that the branch-and-bound algorithm was effective in solving small instances while the heuristic approach was able to produce high-quality solutions for larger instances.

Our contributions. Despite being well studied in the point of view of heuristics, the CSP still lacks effective exact methods. Many of the current best-known solutions still had not been proven optimal or had an open optimality gap due to the absence of a dual bound. The first branch-and-cut framework is proposed for the CSP to address this matter. The framework employs exact and heuristic routines to separate families of valid inequalities, some from the GTSP and others original for the CSP. Computational experiments performed on a benchmark set of instances compare our methodology with the state-of-the-art exact methodology from literature. Previous to this work, from a set of 48 instances, for only 9 instances there were proven optimal solutions. Our methodology improves this by certifying optimality for all except one instance. This represents a major contribution to the current body of knowledge regarding exact approaches on the CSP.

This paper is organized as follows. Section 2 formally defines the CSP and presents an integer linear programming formulation. Section 3 shows new valid inequalities for the CSP. Section 4 describes separation routines for the proposed valid inequalities, which constitute the branch-and-cut framework. In Section 5 computational experiments are conducted on a benchmark of instances, and results are analyzed and discussed. Section 6 gives the concluding remarks.

2. Problem description and formulation

The CSP can be formally stated as follows. Consider an undirected graph $G(V, E)$, where V is the set of vertices and E is the set of edges. Each edge $e \in E$ is associated with a non-negative cost c_e. For each vertex $v \in V$, $C(v)$ is the set of vertices that cover v and $D(v)$ is the set of vertices that are covered by v. It is considered that $v \in C(v)$ and $v \in D(v)$, $\forall v \in V$. An optimal solution to the CSP is a minimum length Hamiltonian cycle (tour) over a subset of vertices that covers all vertices in G. Figures 1a–1c show optimal solutions for three CSP instances with 200 vertices.

An integer linear programming (ILP) formulation for the CSP is presented. Binary variable x_e indicates if an edge $e \in E$ belongs (1) or not (0) to the tour and binary variable y_v represents if a vertex belongs (1) or not (0) to the tour. We denote $\delta(v)$ the set of edges incident to $v \in V$, $\delta(S)$ the set of edges with one endpoint in $S \subset V$ and the other in $V \backslash S$ and $E(S)$ the set of edges with both endpoints in S.

(CSP)

$$\text{MIN} \sum_{e \in E} c_e x_e, \tag{1}$$

subject to

$$\sum_{e \in \delta(v)} x_e = 2y_v \qquad \forall v \in V, \tag{2}$$

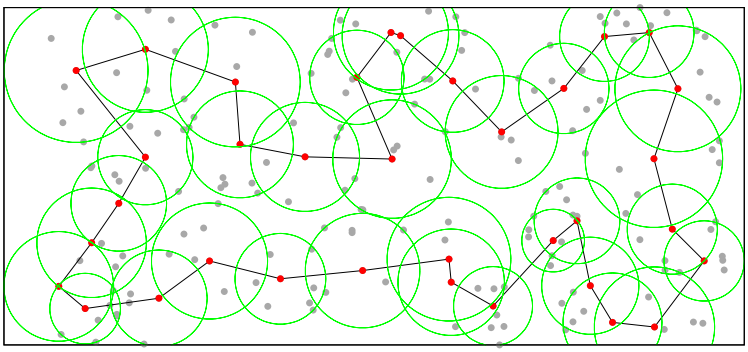
$$\sum_{i \in C(v)} y_i \geqslant 1 \qquad \forall v \in V, \tag{3}$$

$$\sum_{e \in \delta(S)} x_e \geqslant 2(y_i + y_j - 1) \qquad \forall S \subset V, i \in S, j \in V \backslash S, \tag{4}$$

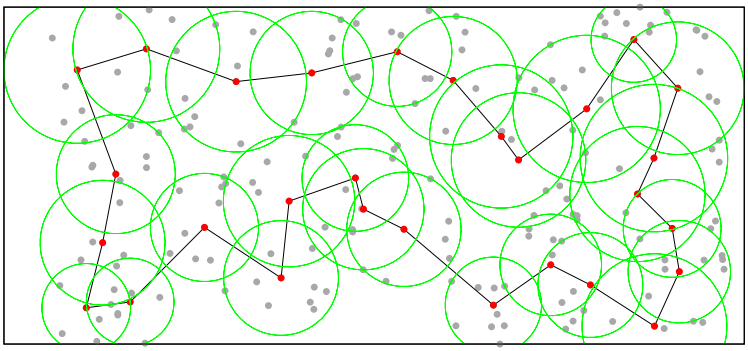
$$x_e \in \{0, 1\} \qquad \forall i, j \in V, \tag{5}$$

$$y_v \in \{0, 1\} \qquad \forall i \in V. \tag{6}$$

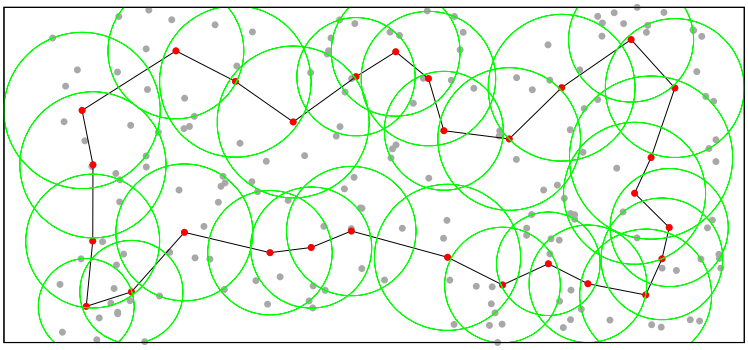
The CSP formulation is based on the ideas of Fischetti *et al.* [12] for the GTSP. The objective function (1) minimizes the cost of a solution given by the sum of the costs of its edges. Constraints (2) ensure the number of edges incident at a vertex is 2 (if v is in the tour) or 0 (otherwise). Constraint (3) impose that each vertex must be covered at least once. Constraint (4) are subtour elimination constraints which state that every cut separating two vertices in the tour contains at least two edges.



(A)



(B)



(C)

FIGURE 1. Optimal solutions for instances kroB200-7, kroB200-9, and kroB200-11, where each vertex covers its closest 7, 9, and 11 neighbors, respectively. Highlighted vertices belong to the tour and their covering sets are represented by circumferences. (a) Instance kroB200-7. (b) Instance kroB200-9. (c) Instance kroB200-11.

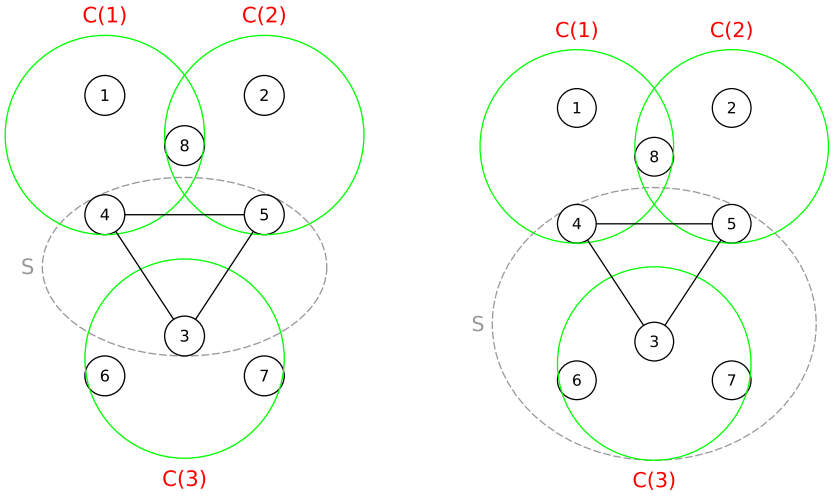


FIGURE 2. Example of $S \notin \gamma(V)$ (a) and $S \in \gamma(V)$ (b).

3. VALID INEQUALITIES

This section presents valid inequalities proposed by Fischetti *et al.* [12] for the GTSP, and here translated for the CSP. It is worth reminding that the GTSP is a special case of the CSP in which the vertices are partitioned into clusters, and each cluster is formed by vertices which mutually cover themselves, *i.e.*, any two vertices u and v from the same cluster would have $C(u) = C(v)$.

Let $D(S)$ be the union of sets $D(v)$ for all $v \in S$, *i.e.*, $D(S) = \bigcup_{v \in S} D(v)$ and let $\gamma(V)$ be the family of all the subsets S of vertices that contains $C(v)$ for at least one vertex $v \in S$, *i.e.*, $\gamma(V) = \{F \subseteq \mathcal{P}(V) : \forall S \in F, \exists v \in S, C(v) \subseteq S\}$ where $\mathcal{P}(V)$ is the power set of V. To exemplify the concept of $\gamma(V)$, consider sets $C(1) = \{1, 4, 8\}$, $C(2) = \{2, 5, 8\}$ and $C(3) = \{3, 6, 7\}$ as shown in Figure 2. As exemplified in Figure 2a, if $S = \{3, 4, 5\}$, then none of the sets $C(1)$, $C(2)$ and $C(3)$ is a subset of S, thus $S \notin \gamma(V)$. However, if $S = \{3, 4, 5, 6, 7\}$, then set $C(3)$ is contained in S, thus $S \in \gamma(V)$, as shown in Figure 2b. The following family of inequalities are valid for the CSP:

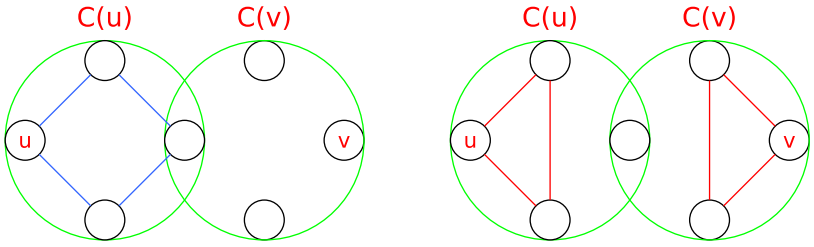
$$\sum_{e \in \delta(S)} x_e \geqslant 2 \qquad \forall S \in \gamma(V) : D(S) \neq V, \tag{7}$$

$$\sum_{e \in \delta(S)} x_e \geqslant 2y_i \qquad \forall S \notin \gamma(V) : D(S) \neq V, i \in S, \tag{8}$$

$$\sum_{e \in \delta(S)} x_e \geqslant 2(y_i + y_j - 1) \qquad \forall S \notin \gamma(V) : D(S) = V, i \in S, j \in V \backslash S. \tag{9}$$

Inequalities (7) ensure that each cut separating two sets $C(v)$ and $C(w)$ must be crossed at least twice. Inequalities (8) imply that each cut separating one vertex in the tour and one set $C(v)$ must be crossed at least twice. Inequalities (9) ensure that each cut separating two vertices in the tour must be crossed at least twice. Originally in GTSP, inequalities (7)–(9) were applied to every subset of vertices containing at least one cluster, *i.e.*, any subset of family $\gamma(V)$.

In the following, a new family of valid inequalities is proposed to consider a scenario particular to the CSP.



(A) (B)

FIGURE 3. Example of feasible (a) and infeasible (b) solutions in the context of overlap of covering sets.

3.1. Cover intersection inequalities

Consider the case in which two covering sets $C(v)$ and $C(u)$ strictly overlap for some pair of vertices v and u, *i.e.*, $C(v) \cap C(u) \neq \emptyset$ and $C(v) \neq C(u)$. This is a distinguished scenario for the CSP, and it does not occur on the GTSP since in that problem the clusters are disjoint. The new valid inequalities extend the idea of inequalities (7), in the sense of requiring a minimum weight for any edge cut-set separating two covering sets. However, to address the overlap of covering sets, the new valid inequalities (10) also take into account the edge cut-set weight of the intersection $C(v) \cap C(u)$.

For the following new valid inequalities (10), consider $S_v = S \cap C(v)$ for any $v \in V$. These inequalities are here called *CI inequalities* (cover intersection inequalities), and they only require a proper subset $S \subset V$ such that $S \in \gamma(V)$, which means it can be employed even if $D(S) = V$, another case in which inequalities (7) cannot be employed.

(CI inequalities)

$$\sum_{e \in (\delta(S) \bigcup \delta(S_v))} x_e \geqslant 2 \qquad \forall v \in V, \forall S \subset V : S \in \gamma(V). \tag{10}$$

According to constraints (3), for any given vertex v, at least one vertex of $C(v)$ must be visited by the tour. In other words, for any subset $S \subset V$ such that $S \in \gamma(V)$, the tour must visit S_v or $C(v) \setminus S_v$. If set S does not intersect with $C(v)$, then S_v is empty, and (10) reduces to (7). Otherwise, S_v is not empty, and in this case, to satisfy constraints (3), the solution must contain at least two edges in either $\delta(S_v)$ or $\delta(S \setminus S_v)$. Figures 3a and 3b consider $V = C(u) \cup C(v)$ and $S = C(u)$. In this case, $S \in \gamma(V)$, and S_v contains a single vertex since $|C(u) \cap C(v)| = 1$. In Figure 3a, a feasible solution is presented such that, even though the cut-set $\delta(S)$ is empty (the tour is inside S), the edge cut-set $\delta(S_v)$ contains two edges, guaranteeing that node v is covered. An infeasible solution is presented in Figure 3b such that both edge cut-sets $\delta(S)$ and $\delta(S_v)$ are empty.

4. BRANCH-AND-CUT FRAMEWORK

This section presents the separation routines for inequalities (7)–(10). Sections 4.1 and 4.2 present the separation routines for integer and fractional solutions, respectively. In the following sections, consider $\{\mathbf{x^I}, \mathbf{y^I}\}$ and $\{\mathbf{x^F}, \mathbf{y^F}\}$ as integer and fractional solutions for the CSP formulation without the subcycle elimination constraints (4) but possibly including some of the valid inequalities (7)–(10). Also, let $G^I(V^I, E^I)$ and $G^F(V^F, E^F)$ be the graphs induced by $\{\mathbf{x^I}, \mathbf{y^I}\}$ and $\{\mathbf{x^F}, \mathbf{y^F}\}$, respectively. In G^I, every vertex $v \in V^I$ has a weight y_v, such that $y_v \in \mathbf{y^I}$, and every edge $e \in E^I$ has a cost x_e, such that $x_e \in \mathbf{x^I}$. Similarly, in G^F, every vertex $v \in V^F$ has a weight y_v, such that $y_v \in \mathbf{y^F}$, and every edge $e \in E^F$ has a cost x_e, such that $x_e \in \mathbf{x^F}$.

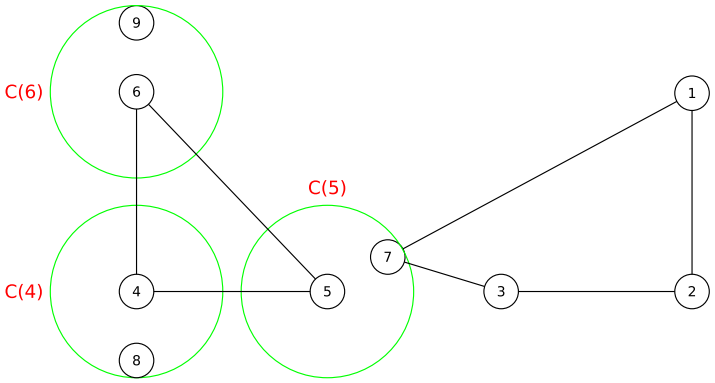


FIGURE 4. Example of an invalid CSP solution with two subcycles.

4.1. Separation routine for integer solutions

The proposed separation routine searches, in a lazy constraint fashion, for inequalities (7)–(10) that are possibly violated by an integer solution $\{\mathbf{x^I}, \mathbf{y^I}\}$. First, the routine performs a depth-first search in G^I to check for the existence of illegal subcycles.

Let $S \subset V$ be the vertices of an illegal subcycle in G^I. To apply inequality (7) or (10) with respect to set S, it is necessary that $S \in \gamma(V)$. If this is not the case, the proposed routine attempts to augment S into S_{aug} by including the set $C(v)$ for some $v \in S$. However, the choice of which $C(v)$ will be included in S_{aug} is relevant to the effectiveness of the corresponding inequalities, as will be explained next.

Consider Figure 4, which shows a solution formed by two subcycles in graph G^I. In this figure $C(v_4) = \{v_4, v_8\}$, $C(5) = \{v_5, v_7\}$, and $C(v_6) = \{v_6, v_9\}$. By taking the illegal subcycle represented by $S = \{v_4, v_5, v_6\}$, it is not possible to apply inequalities (7) or (10), since $S \notin \gamma(V)$. By taking $S_{\text{aug}} = S \cup C(5)$, then $S_{\text{aug}} \in \gamma(V)$, however S_{aug} would not generate an effective cut, since vertex $v_7 \in V^I$. Otherwise, effective cuts can be derived from $S_{\text{aug}} = S \cup C(v_4)$ or $S_{\text{aug}} = S \cup C(v_6)$. Therefore, for an inequality (7) or (10) associated with S_{aug} to be effective in cutting solution $\{\mathbf{x^I}, \mathbf{y^I}\}$, set S_{aug} cannot contain any vertex in $V^I \setminus S$.

Algorithm 1 presents the implementation details of the separation routine for integer solutions, which searches for inequalities (7)–(10) associated with each subcycle found in $\{\mathbf{x^I}, \mathbf{y^I}\}$. The overall complexity of Algorithm 1 is bounded by $O(V^2)$.

4.2. Exact separation routine for fractional solutions

This section gives the exact separation routines of inequalities (7)–(10) for a fractional CSP solution $\{\mathbf{x^F}, \mathbf{y^F}\}$. In particular, the separation of inequalities (7)–(9) follows the methodology proposed by Fischetti *et al.* [12] for the GTSP. As for the CI inequalities (10), a transformation of the solution graph G' is proposed to tackle the overlap of covering sets. The routines are described next.

As observed by Fischetti *et al.* [12], the separation problem to find one or more inequalities (9) violated by $\{\mathbf{x^F}, \mathbf{y^F}\}$ can be reduced to the problem of computing a minimum cut between two vertices i and j in graph G^F, $i \in S$ and $j \in V^F \setminus S$, *i.e.*, finding the maximum flow from i to j [1]. Similarly, the separation of inequalities (8) can be reduced to computing a minimum cut in graph G^F that separates $i \in S$ and $C(u) \subseteq V^F \setminus S$. In other words, finding the maximum flow from i to t [1], where t is an artificial vertex connected to each $j \in C(u)$ through edges with infinite capacity. As for inequalities (7), the separation problem can be reduced to computing a minimum cut between covering sets $C(v)$ and $C(u)$ in graph G^F, with $C(v) \subseteq S$, $C(u) \subseteq V \setminus S$, and $C(v) \cap C(u) = \emptyset$. A maximum flow from s to t can be computed, such that s and t are artificial vertices connected, respectively, to each vertex in $C(v)$ and $C(u)$ with infinite capacity edges, as illustrated in Figure 5.

Algorithm 1. Separation routine for integer solutions.

Input: graph $G^I(V^I, E^I)$ induced by an infeasible integer solution $\{\mathbf{x^I}, \mathbf{y^I}\}$ for the CSP.
Output: a set T of valid inequalities that cuts $\{\mathbf{x^I}, \mathbf{y^I}\}$.

for each subcycle S in $\{\mathbf{x^I}, \mathbf{y^I}\}$ **do**
  $T \leftarrow \emptyset$
  if $D(S) \neq V$ **then**
    if $S \in \gamma(V)$ **then**
      $T \leftarrow T \cup$ inequality (7) associated with S
    else
      $T \leftarrow T \cup$ inequality (8) associated with S
      for each $v \in S$ **do**
        $S_{\text{aug}} \leftarrow S \cup C(v)$
        if $S_{\text{aug}} \cap (V^I \setminus S) = \emptyset$ **then**
          if $D(S_{\text{aug}}) \neq V$ **then**
            $T \leftarrow T \cup$ inequality (7) associated with S_{aug}
          else
            for $u \in V$ **do**
              $S_u \leftarrow S \cap C(u)$;
              if $\delta(S_u) \cap E^I = \emptyset$ **then**
                $T \leftarrow T \cup$ CI inequality (10) associated with S_{aug} and S_u
  else
    for each subcycle S' in $\{\mathbf{x^I}, \mathbf{y^I}\} : S' \neq S$ **do**
      $T \leftarrow T \cup$ inequality (9) associated with S and S'
return T;

It is worth noting that the separation of inequality (7) does not work when $C(v)$ and $C(u)$ overlap, since every cut separating s and t has infinite weight, as exemplified in Figure 6a.

An exact separation algorithm for CI inequalities (10) is proposed to accomodate the case when two covering sets $C(v)$ and $C(u)$ overlap. The first step is to augment graph G^F, by including an artificial vertex w' and an artificial edge (w, w') for every vertex $w \in C(v) \cap C(u)$. Vertex w is removed from $C(u)$ and vertex w' is included into $C(u)$. Finally, for each $w \in C(v) \cap C(u)$, let T_w be the set of edges with one endpoint being w and the other is in $V \setminus C(v)$. The edges of T_w are excluded from G^F and their total weight is transferred to the artificial edge (w, w'). This ensures that every artificial edge will be counted for in any minimum cut, in the sense that every edge in T_w contributes in their purpose of connecting both covering sets $C(v)$ and $C(u)$, as expected in a feasible solution. Figure 6 illustrates the augmentation of graph G^F.

The separation of a CI inequality (10) reduces to computing a minimum cut between sets $C(v)$ and $C(u)$ in the augmented graph. Let $\delta(S_{\min})$ be the minimum cut between $C(v)$ and $C(u)$ and $S_u = S_{\min} \cap C(u)$. If $\sum_{e \in \delta(S_{\min}) \bigcup \delta(S_u)} x_e$ has a value less than 2, then a violated CI inequality (10) was found.

Algorithm 2 presents the implementation of exact separation routine for fractional solutions. The separation consists in computing a max-flow for each pair of vertices, thus considering a push-relabel algorithm [1] to solve max-flow, the time complexity of Algorithm 2 is bounded by $O(V^4 E)$.

Given the computational effort required for the exact separation of fractional solutions, two alternatives were investigated. The first is based on a *first-found* policy, which follows the same steps of Algorithm 2, however the execution is interrupted once the first inequality which surpasses a given violation threshold ϵ is found. For example, with respect to inequalities (7), given a vertex $v \in V$ and a set $S \in \gamma(V) : D(S) \neq V$, if the following holds, $(2 - \sum_{e \in \delta(S)} x_e > \epsilon)$, then the cut is included in the model and Algorithm 2 halts. The same goes for inequalities (8)–(10).

The second alternative for the exact separation routines resides in the heuristic separation of inequalities (7)–(10), described in the following section.

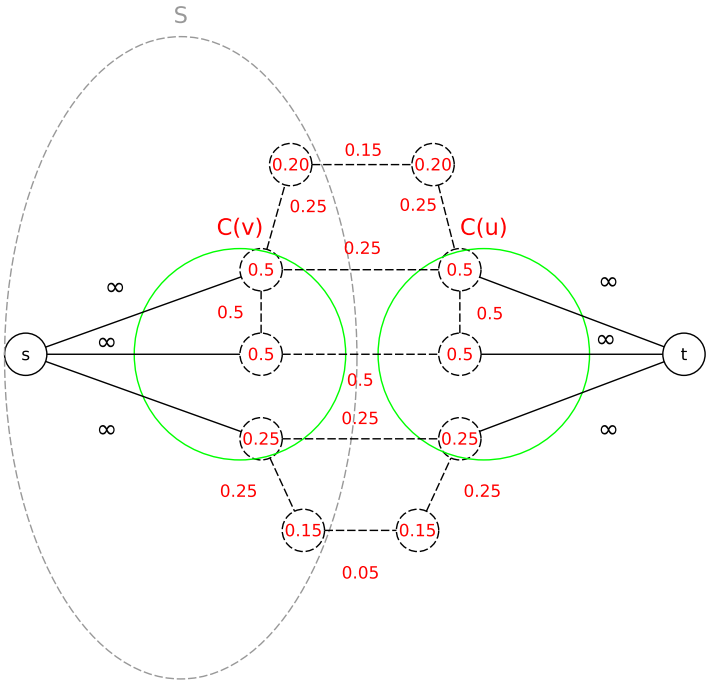


FIGURE 5. Max-flow instance for the separation of inequality (7) in the case where $C(v) \cap C(u) = \emptyset$.

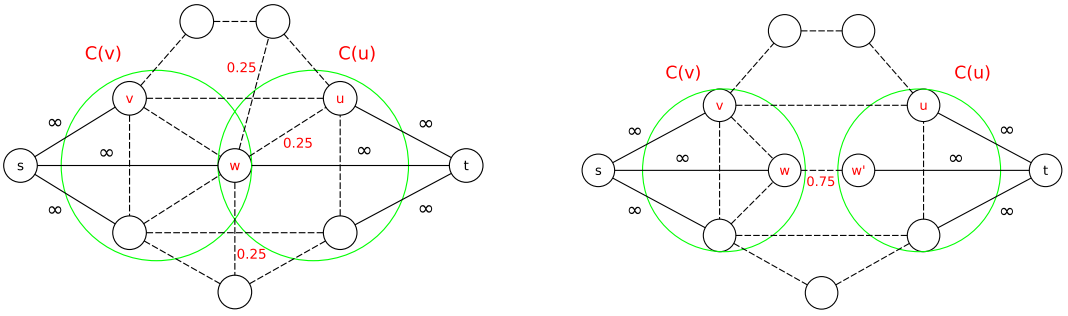


FIGURE 6. Graph augmentation for the separation of CI inequalities (10). (a) Pre-augmentation (b) Augmented graph.

4.3. Heuristic separation routine for fractional solutions

A heuristic separation has the purpose of finding inequalities being violated by a fractional solution $\{\mathbf{x}^{\mathbf{F}}, \mathbf{y}^{\mathbf{F}}\}$ within short computational times. In contrast with the exact separation routine however, a heuristic does not come with any guarantee of finding a violated inequality even if one exists.

The heuristic separation routine for inequalities (7)–(10) is composed of four main steps. The first step searches for inequalities (7) and (10) for every $u \in V$ and its corresponding covering set $C(u)$. In more details, let $S = C(u)$ and consider two cases: (i) if $D(S) \neq V$ and $\sum_{e \in \delta(S)} x_e < 2$, then the inequality (7) associated

Algorithm 2. Exact separation routine for fractional solutions.

Input: graph $G^F(V^F, E^F)$ induced by a fractional solution $\{\mathbf{x^F}, \mathbf{y^F}\}$ for the CSP.
Output: a set T of valid inequalities that cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

$T \leftarrow \emptyset$
for $v \in V$ **do**
  for $u \in V \setminus \{v\}$ **do**
    if $D(C(v)) \neq V$ **and** $C(v) \cap C(u) = \emptyset$ **then**
      $S \leftarrow \text{minCut}(C(v), C(u), G^F)$
      $T \leftarrow T \cup$ inequality (7) associated with S.
    else
      $G^F_{\text{aug}} \leftarrow \text{augment}(G^F)$
      $S \leftarrow \text{minCut}(C(v), C(u), G^F_{\text{aug}})$
      $T \leftarrow T \cup$ CI inequality (10) associated with S and u.
    if $y_v > 0$ **and** $v \notin C(u)$ **then**
      $S \leftarrow \text{minCut}(v, C(u), G^F)$
      $T \leftarrow T \cup$ inequality (8) associated with S and u.
    if $y_v + y_u - 1 > 0$ **then**
      $S \leftarrow \text{minCut}(v, u, G^F)$
      $T \leftarrow T \cup$ inequality (9) associated with S, v and u.
return T;

with S cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$; (ii) if $D(S) = V$ and $\sum_{e \in \delta(S)} x_e + \sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ for some vertex $v \in V \setminus \{u\}$, then the CI inequality (10) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

In the second step, the connected components $S_1, \ldots, S_p$ of G^F are computed. For each component S_k, let $S = S_k$ and if $S \in \gamma(V)$ then two cases are considered: (i) if $D(S) \neq V$, then the inequality (7) associated with S cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$; (ii) if $D(S) = V$ and $\sum_{e \in \delta(S_v)} x_e < 2$ for some vertex $v \in V$, then the CI inequality (10) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

In the third step, for each connected component S_k, let $S = S_k$ and $i = \arg\max_v \{y_v : v \in S\}$. If $D(S) \neq V$, then the inequality (8) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

Finally, the fourth step iterates through all pairs of connected components S_k and S_l, $k \neq l$. For each pair, let $i = \arg\max_v \{y_v : v \in S_k\}$ and $j = \arg\max_v \{y_v : v \in S_l\}$. If $y_i + y_j > 1$, the inequality (9) associated with $S = S_k$, i, and j cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

Algorithm 3, with a time complexity bounded by $O(V^2)$, details the heuristic separation routine for fractional solutions.

5. Computational experiments

In this section, the proposed branch-and-cut methodologies are evaluated and compared to the state-of-the-art using the literature benchmark of instances, described in the following section.

5.1. Instances

The benchmark used in the computational experiments is composed of instances by Salari *et al.* [20] and a new set of instances, all of them based on the TSPLIB [18]. Salari *et al.* [20] divided the instances into three types: small (36 instances, $51 \leqslant |V| \leqslant 100$), medium (12 instances, $150 \leqslant |V| \leqslant 200$), and large (27 instances, $532 \leqslant |V| \leqslant 783$). The covering set of each vertex is defined by its k closest vertices. For each graph of small and medium instances, three values of k were used, $k = 7$, $k = 9$, and $k = 11$. In order to complement the medium instances, we also created 39 new instances with $225 \leqslant |V| \leqslant 493$ to complement the set from the literature. The large instances were tested in the literature for heuristics only, and for this reason, our results

Algorithm 3. Heuristic separation routine for fractional solutions.

Input: graph $G^F(V^F, E^F)$ induced by a fractional solution $\{\mathbf{x^F}, \mathbf{y^F}\}$ for the CSP.
Output: a set T of valid inequalities that cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

$T \leftarrow \emptyset$
for $u \in V$ **do**
  $S \leftarrow C(u)$;
  if $D(S) \neq V$ **then**
    if $\sum_{e \in \delta(S)} x_e < 2$ **then**
      $T \leftarrow T \cup$ inequality (7) associated with S.
  else
    for $v \in V : v \neq u$ **do**
      $S_v \leftarrow S \cap C(v)$;
      if $\sum_{e \in \delta(S)} x_e + \sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ **then**
        $T \leftarrow T \cup$ CI inequality (10) associated with S and v.
Compute the connected components $S_1, \ldots, S_p$ of G^F;
for $k = 1, ..., p$ **do**
  $S \leftarrow S_k$
  if $S \in \gamma(V)$ **then**
    if $D(S) \neq V$ **then**
      $T \leftarrow T \cup$ inequality (7) associated with S.
    else
      for $v \in V : v \neq w$ **do**
        $S_v \leftarrow S \cap C(v)$;
        if $\sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ **then**
          $T \leftarrow T \cup$ CI inequality (10) associated with S and v.
  else
    $i \leftarrow \arg\max_v \{y_v : v \in S\}$.
    if $D(S) \neq V$ **then**
      $T \leftarrow T \cup$ inequality (8) associated with S and i.
    for $l = k, ..., p$ **do**
      $j \leftarrow \arg\max_v \{y_v : v \in S_l\}$
      $T \leftarrow T \cup$ inequality (9) associated with S, i and j.
return T

for these instances were not included in the paper. Nonetheless, full experimental data (including results for large instances) and source codes are available on-line[1].

5.2. Computational settings

The branch-and-cut methodologies were implemented in C++ using solver Gurobi and the Lemon graph library [9]. The experiments were conducted on a PC under Ubuntu and CPU Intel Xeon E5-2630 2.2 GHz, with 64GB of RAM and one-hour time limit.

5.3. Evaluated methodologies

Five branch-and-cut methodologies were implemented and evaluated in the computational experiments:

(1) CSP-I: exact separation routine for integer solutions (Algorithm 1) considering valid inequalities (7)–(9), but excluding the CI inequalities (10);

[1] http://www.ic.unicamp.br/~fusberti/problems/csp.

(2) CSP-I&F$_{vp}$: uses the same exact separation routine for integer solutions as CSP-I. On the root node, exact separation routine for fractional solutions (Algorithm 2) considering inequalities (7)–(9), but excluding the CI inequalities for CSP (10). For the non-root nodes, Algorithm 2 was implemented under the *first-found* policy with violation threshold $\epsilon = 1$ (see Sect. 4.2).
(3) CSP-I&F$_{vp}$-X: same as CSP-I&F$_{vp}$, but including the CI inequalities (10);
(4) CSP-I&F$_{h}$: uses the same exact separation routine for integer solutions as CSP-I. On the root node, exact separation routine for fractional solutions (Algorithm 2) considering inequalities (7)–(9), but excluding the CI inequalities (10). For the non-root nodes, heuristic separation for fractional solutions (Algorithm 3), considering inequalities (7)–(9), but excluding the CI inequalities (10);
(5) CSP-I&F$_{h}$-X: same as CSP-I&F$_{h}$, but including inequalities (10).

These methodologies were compared with the integer linear programming formulation proposed by Salari *et al.* [20], denoted here as SRS. To the best of our knowledge, SRS is the best performing exact methodology for the CSP.

Preliminary experiments have shown that even in cases where the heuristic separation fails to find violated inequalities in methodologies CSP-I&F$_{h}$ and CSP-I&F$_{h}$-X, applying the exact separation does not improve the quality of the solutions obtained. This can be justified by the high computational effort spent by the exact separation routines.

5.4. Results

The results of the computational experiments are reported for the small instances in Table 2 and for the medium instances in Tables 3 and 4. Each table reports for each methodology and for each instance, the following:

– *LB*: best lower bound obtained;
– *Gap*: optimality gap $(\frac{\text{UB}-\text{LB}}{\text{UB}}) \cdot 100$;
– *Time*: execution time in seconds.

In Tables 2 and 3, the column group *BestUB* reports the best upper bounds known in the literature for each instance: column UB gives the best known upper bounds and column *References* cites the papers which attained them. For each instance, Tables 2–4 highlight the optimal solutions (underlined) and the best lower bounds (in bold) obtained by each methodology.

For small-size instances, there were previously known lower bounds for 32 out of 36 instances, obtained by SRS, from which optimal solutions were proven for 9 instances. The proposed branch-and-cut framework, on the other hand, obtained lower bounds for all instances. More importantly, the framework proved optimality for all 36 small instances. All branch-and-cut methodologies outperformed SRS with respect to optimality gap, and they were fairly robust among themselves; the worst performing (CSP-I) obtained an average 0.28% optimality gap, while the best performing (CSP-I&F$_{vp}$ and CSP-I&F$_{vp}$-X) with zero optimality gap, shows the exact separation prevails over the heuristic separation of fractional solutions for small instances.

With respect to medium-size instances created by Salari *et al.*, no lower bound was known for any of the 12 instances in the literature. The branch-and-cut framework obtained the first lower bounds for all these instances. Furthermore, optimality was proven for all instances except one (kroA200-7), which remains with an optimality gap of 1.35%. The performance among the branch-and-cut methodologies varied more significantly this time. The best-performing methodology was CSP-I&F$_{h}$-X, with an average gap of 0.11%. The heuristic separation overcomes the exact separation, mainly due to the reduction in computational effort. The worst-performing methodology (CSP-I) obtained an average gap of 6.12%, showing that integral cuts alone perform poorly for more challenging instances.

Regarding the medium-size instances generated in this work, the proposed branch-and-cut framework obtained lower bounds for all instances. Among the 39 instances, the framework proved optimality for 2 instances. The best methodology was CSP-I&F$_{h}$-X, with an average lower bound of 20 419, followed by methodologies

TABLE 2. Results of computational experiments for the small-size instances.

		BestUB		SRS			CSP-I			CSP-I&F$_{vp}$			CSP-I&F$_{h}$			CSP-I&F$_{vp}$-X			CSP-I&F$_{h}$-X		
Instance	NC	UB	References	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
eil51	7	164	[14, 16, 17, 19, 20, 24, 25]	**164**	**0**	149	**164**	**0**	2	**164**	**0**	4	**164**	**0**	3	**164**	**0**	3	**164**	**0**	3
	9	159	[14, 16, 17, 19, 20, 24, 25]	**159**	**0**	220	**159**	**0**	1	**159**	**0**	2	**159**	**0**	2	**159**	**0**	4	**159**	**0**	3
	11	147	[14, 16, 17, 19, 20, 24, 25]	**147**	**0**	681	**147**	**0**	1	**147**	**0**	2	**147**	**0**	2	**147**	**0**	5	**147**	**0**	4
berlin52	7	3887	[14, 16, 17, 19, 20, 24, 25]	**3887**	**0**	140	**3887**	**0**	2	**3887**	**0**	2	**3887**	**0**	2	**3887**	**0**	4	**3887**	**0**	3
	9	3430	[14, 16, 17, 19, 20, 24, 25]	**3430**	**0**	212	**3430**	**0**	1	**3430**	**0**	3	**3430**	**0**	2	**3430**	**0**	6	**3430**	**0**	3
	11	3262	[14, 16, 17, 19, 20, 24, 25]	**3262**	**0**	255	**3262**	**0**	1	**3262**	**0**	2	**3262**	**0**	2	**3262**	**0**	4	**3262**	**0**	4
st70	7	288	[14, 16, 17, 19, 20, 24, 25]	**288**	**0**	490	**288**	**0**	3	**288**	**0**	7	**288**	**0**	5	**288**	**0**	7	**288**	**0**	7
	9	259	[14, 16, 17, 19, 20, 24, 25]	**259**	**0**	1391	**259**	**0**	3	**259**	**0**	7	**259**	**0**	6	**259**	**0**	13	**259**	**0**	9
	11	247	[14, 16, 17, 19, 20, 24, 25]	218	13.14	3600	**247**	**0**	3	**247**	**0**	7	**247**	**0**	5	**247**	**0**	14	**247**	**0**	9
eil76	7	207	[14, 16, 17, 19, 20, 24]	193	7.45	3600	**207**	**0**	6	**207**	**0**	29	**207**	**0**	9	**207**	**0**	15	**207**	**0**	12
	9	185	[16, 17, 19, 25]	161	14.65	3600	**185**	**0**	10	**185**	**0**	10	**185**	**0**	9	**185**	**0**	13	**185**	**0**	12
	11	170	[14, 16, 17, 19, 20, 24, 25]	145	17.08	3600	**170**	**0**	6	**170**	**0**	8	**170**	**0**	7	**170**	**0**	15	**170**	**0**	13
pr76	7	50 275	[14, 16, 17, 19, 20, 24, 25]	**50 275**	**0**	2488	**50 275**	**0**	7	**50 275**	**0**	8	**50 275**	**0**	6	**50 275**	**0**	12	**50 275**	**0**	8
	9	45 348	[14, 16, 17, 19, 20, 24, 25]	42 935	5.62	3600	**45 348**	**0**	6	**45 348**	**0**	32	**45 348**	**0**	10	**45 348**	**0**	16	**45 348**	**0**	14
	11	43 028	[14, 16, 17, 19, 20, 24, 25]	39 022	10.27	3600	**43 028**	**0**	28	**43 028**	**0**	17	**43 028**	**0**	46	**43 028**	**0**	27	**43 028**	**0**	28
rat99	7	486	[14, 16, 17, 19, 20, 24, 25]	433	12.21	3600	**486**	**0**	238	**486**	**0**	19	**486**	**0**	17	**486**	**0**	33	**486**	**0**	17
	9	455	[14, 16, 17, 19, 20, 24, 25]	377	20.73	3600	438	3.88	3600	**455**	**0**	30	**455**	**0**	27	**455**	**0**	28	**455**	**0**	29
	11	444	[14, 16, 17, 19, 20, 24, 25]	350	26.81	3600	**444**	**0**	203	**444**	**0**	32	**444**	**0**	138	**444**	**0**	37	**444**	**0**	197
kroA100	7	9674	[14, 16, 17, 19, 20, 24, 25]	9177	5.42	3600	**9674**	**0**	15	**9674**	**0**	18	**9674**	**0**	27	**9674**	**0**	27	**9674**	**0**	30
	9	9159	[14, 16, 17, 19, 20, 24, 25]	7938	15.38	3600	**9159**	**0**	154	**9159**	**0**	28	**9159**	**0**	2040	**9159**	**0**	36	**9159**	**0**	2230
	11	8901	[14, 16, 17, 19, 20, 24, 25]	8593	3.59	3600	8608	3.40	3600	**8901**	**0**	45	8640	3.02	3600	**8901**	**0**	79	8801	1.14	3600
kroB100	7	9537	[14, 16, 17, 19, 20, 24, 25]	–	–	3600	**9537**	**0**	45	**9537**	**0**	22	**9537**	**0**	20	**9537**	**0**	26	**9537**	**0**	23
	9	9240	[14, 16, 17, 19, 20, 24, 25]	7678	20.34	3600	**9240**	**0**	363	**9240**	**0**	21	**9240**	**0**	21	**9240**	**0**	31	**9240**	**0**	28
	11	8842	[14, 16, 17, 19, 20, 24, 25]	–	–	3600	**8842**	**0**	141	**8842**	**0**	25	**8842**	**0**	29	**8842**	**0**	40	**8842**	**0**	36
kroC100	7	9723	[14, 16, 17, 19, 20, 24, 25]	8564	13.54	3600	**9723**	**0**	561	**9723**	**0**	107	**9723**	**0**	102	**9723**	**0**	67	**9723**	**0**	92
	9	9171	[14, 16, 17, 19, 20, 24, 25]	7663	19.68	3600	8920	2.81	3600	**9171**	**0**	45	**9171**	**0**	783	**9171**	**0**	123	**9171**	**0**	972
	11	8632	[14, 16, 17, 19, 20, 24, 25]	7590	13.73	3600	**8632**	**0**	254	**8632**	**0**	38	**8632**	**0**	820	**8632**	**0**	222	**8632**	**0**	870
kroD100	7	9626	[14, 16, 17, 19, 20, 24, 25]	8724	10.34	3600	**9626**	**0**	59	**9626**	**0**	17	**9626**	**0**	20	**9626**	**0**	34	**9626**	**0**	23
	9	8885	[14, 16, 17, 19, 20, 24, 25]	–	–	3600	**8885**	**0**	16	**8885**	**0**	22	**8885**	**0**	27	**8885**	**0**	62	**8885**	**0**	35
	11	8725	[14, 16, 17, 19, 20, 24, 25]	–	–	3600	**8725**	**0**	51	**8725**	**0**	35	**8725**	**0**	48	**8725**	**0**	63	**8725**	**0**	80
kroE100	7	10 150	[14, 16, 17, 19, 20, 24, 25]	9274	9.44	3600	**10 150**	**0**	520	**10 150**	**0**	81	**10 150**	**0**	42	**10 150**	**0**	76	**10 150**	**0**	32
	9	8991	[14, 16, 17, 19, 24, 25]	8500	5.77	3600	**8991**	**0**	336	**8991**	**0**	31	**8991**	**0**	55	**8991**	**0**	28	**8991**	**0**	88
	11	8450	[14, 16, 17, 19, 20, 24, 25]	7739	9.19	3600	**8450**	**0**	237	**8450**	**0**	23	**8450**	**0**	261	**8450**	**0**	33	**8450**	**0**	193
rd100	7	3461	[14, 16, 17, 19, 20, 24, 25]	3094	11.88	3600	**3461**	**0**	119	**3461**	**0**	20	**3461**	**0**	20	**3461**	**0**	24	**3461**	**0**	22
	9	3194	[14, 16, 17, 19, 20, 24, 25]	2664	19.90	3600	**3194**	**0**	63	**3194**	**0**	18	**3194**	**0**	18	**3194**	**0**	24	**3194**	**0**	25
	11	2922	[14, 16, 17, 19, 20, 24, 25]	2648	10.33	3600	**2922**	**0**	28	**2922**	**0**	21	**2922**	**0**	20	**2922**	**0**	34	**2922**	**0**	27
Avg				**7673.50**	**9.27**	**2867.39**	**8310.08**	**0.28**	**396.75**	**8325.67**	**0.00**	**23.28**	**8318.42**	**0.08**	**229.19**	**8325.67**	**0.00**	**35.69**	**8322.89**	**0.03**	**243.92**

TABLE 3. Results of computational experiments for the medium-size instances.

		BestUB		CSP-I			CSP-I&F$_{vp}$			CSP-I&F$_{h}$			CSP-I&F$_{vp}$-X			CSP-I&F$_{h}$-X		
Instance	NC	UB	Reference(s)	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
kroA150	7	11 423	[14, 16, 17, 19, 20, 24, 25]	10 658	7.18	3600	**11 423**	**0**	174	**11 423**	**0**	137	**11 423**	**0**	147	**11 423**	**0**	90
	9	10 056	[14, 16, 17, 19, 20, 24, 25]	**10 056**	**0**	147	**10 056**	**0**	84	**10 056**	**0**	85	**10 056**	**0**	92	**10 056**	**0**	122
	11	9439	[14, 16, 17, 19, 20, 24, 25]	9240	2.15	3600	**9439**	**0**	95	**9439**	**0**	67	**9439**	**0**	243	**9439**	**0**	91
kroB150	7	11 457	[14, 16, 17, 19, 20, 24, 25]	10 663	7.45	3600	**11 457**	**0**	334	**11 457**	**0**	116	**11 457**	**0**	113	**11 457**	**0**	81
	9	10 121	[14, 16, 17, 19, 20, 24, 25]	9951	1.71	3600	**10 121**	**0**	280	**10 121**	**0**	130	**10 121**	**0**	145	**10 121**	**0**	112
	11	9611	[14, 16, 17, 19, 20, 24, 25]	**9611**	**0**	902	**9611**	**0**	849	**9611**	**0**	429	**9611**	**0**	947	**9611**	**0**	282
kroA200	7	13 285	[14, 16, 19, 25]	11 660	13.94	3600	12 611	5.34	3600	12 955	2.55	3600	12 697	4.63	3600	**13 108**	1.35	3600
	9	11 708	[14, 16, 17, 19, 20, 24, 25]	10 327	13.37	3600	11 094	5.53	3600	**11 708**	**0**	2252	11 537	1.48	3600	**11 708**	**0**	1008
	11	10 748	[14, 16, 17, 19, 25]	9508	13.04	3600	10 342	3.93	3600	**10 748**	**0**	1044	**10 748**	**0**	3582	**10 748**	**0**	648
kroB200	7	13 051	[14, 16, 17, 19, 20, 24, 25]	12 260	6.45	3600	12 462	4.73	3600	12 904	1.14	3600	12 697	2.79	3600	**13 051**	**0**	1487
	9	11 864	[16, 17, 19, 20, 24, 25]	11 209	5.84	3600	11 379	4.26	3600	**11 864**	**0**	2281	11 695	1.45	3600	**11 864**	**0**	1242
	11	10 644	[16, 17, 19, 20, 24, 25]	10 405	2.30	3600	**10 644**	**0**	800	**10 644**	**0**	907	**10 644**	**0**	514	**10 644**	**0**	938
Avg				**10 462.33**	**6.12**	**3087.42**	**10 886.58**	**1.98**	**1718.00**	**11 077.50**	**0.31**	**1220.67**	**11 010.50**	**0.86**	**1681.92**	**11 102.42**	**0.11**	**808.42**

TABLE 4. Results of computational experiments for the new medium-size instances created.

		CSP-I			CSP-I&F$_{vp}$			CSP-I&F$_h$			CSP-I&F$_{vp}$-X			CSP-I&F$_h$-X		
Instance	NC	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
	7	68 805	3.74	3600	70 254	1.79	3600	**70 988**	0.50	3600	70 802	0.83	3600	70 976	0.25	3600
ts225	9	59 718	10.36	3600	61 834	16.17	3600	64 048	1.72	3600	62 899	4.77	3600	**64 404**	1.22	3600
	11	50 022	15.54	3600	51 057	15.51	3600	53 691	8.45	3600	52 755	37.07	3600	**53 786**	7.40	3600
	7	1565	8.05	3600	1622	3.70	3600	1649	1.88	3600	1644	2.19	3600	**1669**	0.66	3600
tsp225	9	1310	15.34	3600	1476	2.30	3600	1492	1.21	3600	1500	0.67	3600	**1510**	**0.00**	1218
	11	1223	12.26	3600	1310	6.49	3600	1346	1.56	3600	1340	2.01	3600	**1367**	**0.00**	1271
	7	**50 479**	13.89	3600	47 279	–	3600	50 216	85.87	3600	48 646	–	3600	50 420	89.03	3600
pr226	9	**50 916**	8.90	3600	47 230	–	3600	49 911	–	3600	51 110	43.43	3600	50 191	85.38	3600
	11	**50 236**	6.51	3600	44 162	–	3600	48 925	–	3600	45 569	–	3600	49 444	127.37	3600
	7	938	12.15	3600	931	–	3600	**991**	6.76	3600	943	–	3600	965	–	3600
gil262	9	804	20.27	3600	865	–	3600	888	–	3600	877	–	3600	**901**	5.33	3600
	11	798	10.03	3600	850	3.06	3600	850	2.71	3600	**856**	1.99	3600	854	2.69	3600
	7	19 057	191.89	3600	20 360	–	3600	20 970	–	3600	20 425	–	3600	**20 983**	–	3600
pr264	9	17 962	–	3600	18 986	–	3600	20 257	–	3600	19 115	–	3600	**20 497**	375.64	3600
	11	17 052	–	3600	18 790	–	3600	**19 748**	409.25	3600	19 183	–	3600	19 556	–	3600
	7	772	53.63	3600	1046	–	3600	1083	21.79	3600	**1088**	122.15	3600	1086	–	3600
a280	9	876	25.23	3600	946	15.22	3600	**1003**	2.89	3600	988	5.87	3600	983	–	3600
	11	817	20.93	3600	871	12.40	3600	911	5.16	3600	917	–	3600	**939**	2.56	3600
	7	17 263	42.72	3600	20 545	–	3600	21 488	16.31	3600	21 291	16.21	3600	**21 517**	–	3600
pr299	9	18 121	20.46	3600	19 609	–	3600	20 092	5.14	3600	19 972	143.62	3600	**20 305**	3.97	3600
	11	13 806	45.84	3600	18 256	–	3600	**18 485**	5.17	3600	18 275	23.76	3600	18 452	5.84	3600
	7	17 634	23.90	3600	18 715	12.95	3600	**19 331**	–	3600	18 991	209.48	3600	18 886	–	3600
lin318	9	14 073	42.85	3600	16 012	–	3600	16 718	–	3600	16 229	223.08	3600	**16 920**	15.76	3600
	11	12 389	48.97	3600	15 511	–	3600	15 425	247.70	3600	**15 809**	–	3600	15 526	236.91	3600
	7	4272	73.36	3600	5935	–	3600	**6093**	–	3600	6054	–	3600	6041	–	3600
rd400	9	3705	82.73	3600	5354	–	3600	5342	237.06	3600	**5389**	–	3600	5388	–	3600
	11	2948	93.69	3600	4755	–	3600	4749	22.53	3600	4819	–	3600	**4825**	21.24	3600
	7	4670	141.31	3600	5465	–	3600	5452	1125.84	3600	**5553**	–	3600	5540	–	3600
fl417	9	3900	246.90	3600	5011	–	3600	**5447**	–	3600	5097	–	3600	5281	–	3600
	11	4119	–	3600	**4894**	–	3600	4882	–	3600	4891	–	3600	4881	–	3600
	7	38 920	81.77	3600	49 869	–	3600	49 794	–	3600	**50 865**	–	3600	50 704	–	3600
pr439	9	35 796	121.95	3600	42 732	–	3600	42 734	–	3600	**46 196**	–	3600	46 046	–	3600
	11	30 738	85.65	3600	43 008	–	3600	42 962	369.00	3600	**43 389**	–	3600	43 321	–	3600
	7	13 992	89.07	3600	19 760	–	3600	19 768	20.62	3600	**2 0618**	–	3600	20 560	–	3600
pcb442	9	12 390	142.28	3600	17 009	–	3600	17 105	–	3600	**18 434**	–	3600	18 403	35.18	3600
	11	10 760	103.79	3600	16 436	–	3600	16 144	–	3600	**17 478**	13.13	3600	17 448	16.94	3600
	7	12 961	91.00	3600	16 160	–	3600	16 044	–	3600	**16 628**	–	3600	16 373	–	3600
d493	9	11 939	78.78	3600	15 010	–	3600	14 796	–	3600	15 078	–	3600	**15 230**	–	3600
	11	10 935	92.91	3600	13 937	–	3600	13 894	–	3600	**14 189**	18.40	3600	14 146	–	3600
Avg		**17 658**		**3600**	**19 586**		**3600**	**20 146**		**3600**	**20 151**		**3600**	**20 419**		**3479**

CSP-I&F$_{vp}$-X, CSP-I&F$_h$ and CSP-I&F$_{vp}$ with average lower bounds of 20 151, 20 146, and 19 586, respectively. Moreover, CSP-I&F$_h$-X obtained the best lower bound among all methodologies for 14 instances, followed by CSP-I&F$_{vp}$-X with 13 instances, CSP-I&F$_h$ with 8 instances, and CSP-I&F$_{vp}$ with 1 instance.

The effect of the CI inequalities (10) in the performance of the methodologies was also examined. Regarding the small and medium instances created by Salari *et al.*, the average gaps of CSP-I&F$_{vp}$ and CSP-I&F$_{vp}$-X were both zero for small instances, but for the medium instances the CI inequalities reduced the average gap

from 1.98% to 0.86%. Furthermore, comparing CSP-I&F_h and CSP-I&F_h-X, the CI inequalities reduced the average gaps from 0.08% to 0.03% for small instances and from 0.31% to 0.11% for medium instances. It is worth mentioning that even for the medium instances where the methodologies CSP-I&F_{vp} and CSP-I&F_h obtained the same optimality gaps of CSP-I&F_{vp}-X and CSP-I&F_h-X, the CI inequalities reduced the execution time on average. For example, comparing the methodologies that use heuristic separation, the difference in execution time was substantial. The average execution times of CSP-I&F_h and CSP-I&F_h-X were 1220.67 and 808.42, respectively, representing a reduction of more than 33%.

Considering the new medium instances, from a total of 39 instances, CSP-I&F_{vp}-X and CSP-I&F_h-X obtained, together, the best lower bounds for 27 instances, while CSP-I&F_{vp} and CSP-I&F_h obtained the best lower bounds for only 9 instances. Furthermore, only CSP-I&F_h-X was able to obtain optimal solutions. Therefore, the results show that the CI inequalities have a significant impact on reducing the optimality gaps and obtaining the best lower bounds.

6. Final remarks

The proposed branch-and-cut framework for the CSP uses existing valid inequalities for the GTSP, by Fischetti *et al.* [12], and a new family of valid inequalities, *CI inequalities*, to improve on the state-of-the-art exact methodology for the CSP. Exact and heuristic separation routines for integer and fractional solutions are investigated.

The branch-and-cut framework is composed of five methodologies using distinct families of inequalities and separation routines. Computational experiments conducted on a benchmark of 48 instances from literature and 39 new instances delves into the effectiveness of the framework. From the 48 small and medium instances from literature, only 9 optimal solutions were known. Our branch-and-cut framework, by borrowing meaningful valid inequalities from GTSP and proposing new valid inequalities for CSP, was able to obtain optimal solutions for all instances except one, thus 47 instances were proven optimal (among them, 38 instances for the first time). With respect to the new instances, our methodology obtained the highest number of best lower bounds and number of proven optimal solutions. Finally, the experiments also show that the CI inequalities had a significant role in the performance of the methodologies.

The ideas presented in this work can support the exact solution of many future developments of the CSP. The heuristics and metaheuristics approaches proposed to solve the CSP in the literature can be combined with our branch-and-cut framework to develop hybrid methodologies, such as matheuristics, in order to improve the upper and lower bounds of the CSP. Future works may consider, for example, CSP with multiple vehicles, capacity constraints, time constraints, green vehicles, uncertainty on the covering neighborhood, and other generalizations of the CSP which better approximate practical routing problems. The new family of valid inequalities proposed in this work should be considered on the exact solution for any of these generalizations.

Acknowledgements. This work was supported by CAPES, CNPq, and Fapesp (grants 140960/2017-1, 314384/2018-9, 435520/2018-0, 2015/11937-9).

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993).

[2] D.L. Applegate, R.E. Bixby, V. Chvatal and W.J. Cook, The Traveling Salesman Problem: A Computational Study. *Princeton Series in Applied Mathematics*. Princeton University Press, Princeton, NJ, USA (2007).

[3] B. Behdani and J.C. Smith, An integer-programming-based approach to the close-enough traveling salesman problem. *INFORMS J. Comput.* **26** (2014) 415–432.

[4] W.P. Coutinho, R.Q.D. Nascimento, A.A. Pessoa and A. Subramanian, A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS J. Comput.* **28** (2016) 752–765.

[5] J.R. Current and D.A. Schilling, The covering salesman problem. *Transp. Sci.* **23** (1989) 208–213.

[6] J.R. Current and D.A. Schilling, The median tour and maximal covering tour problems: formulations and heuristics. *Eur. J. Oper. Res.* **73** (1994) 114–126.

[7] J. Current, H. Pirkul and E. Rolland, Efficient algorithms for solving the shortest covering path problem. *Transp. Sci.* **28** (1994) 317–327.
[8] M. Dell'Amico, R. Montemanni and S. Novellani, Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega* **104** (2021) 102493.
[9] B. Dezső, A. Jüttner and P. Kovács, LEMON – an open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.* **264** (2011) 23–45.
[10] J. Dong, N. Yang and M. Chen, Heuristic approaches for a TSP variant: the automatic meter reading shortest tour problem, in *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer (2007) 145–163.
[11] A. Dumitrescu and J.S.B. Mitchell, Approximation algorithms for tsp with neighborhoods in the plane. *J. Algorithms* **48** (2003) 135–159.
[12] M. Fischetti, J.J.S. González and P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Oper. Res.* **45** (1997) 378–394.
[13] M. Gendreau, G. Laporte and F. Semet, The covering tour problem. *Oper. Res.* **45** (1997) 568–576.
[14] B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari and P. Toth, The generalized covering salesman problem. *INFORMS J. Comput.* **24** (2012) 534–553.
[15] D.J. Gulczynski, J.W. Heath and C.C. Price, The close enough traveling salesman problem: a discussion of several heuristics, in Perspectives in Operations Research. Springer (2006) 271–283.
[16] Y. Lu, U. Benlic and Q. Wu, A highly effective hybrid evolutionary algorithm for the covering salesman problem. *Inf. Sci.* **564** (2021) 144–162.
[17] V. Pandiri, A. Singh and A. Rossi, Two hybrid metaheuristic approaches for the covering salesman problem. *Neural Comput. App.* **32** (2020) 15643–15663.
[18] G. Reinelt, TSPLIB – a traveling salesman problem library. *ORSA J. Comput.* **3** (1991) 376–384.
[19] M. Salari and Z. Naji-Azimi, An integer programming-based local search for the covering salesman problem. *Comput. Oper. Res.* **39** (2012) 2594–2602.
[20] M. Salari, M. Reihaneh and M.S. Sabbagh, Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Comput. Ind. Eng.* **83** (2015) 244–251.
[21] R. Shuttleworth, B.L. Golden, S. Smith and E. Wasil, Advances in meter reading: heuristic solution of the close enough traveling salesman problem over a street network, in The Vehicle Routing Problem: Latest Advances and New Challenges. Springer (2008) 487–501.
[22] G.O. Tiniç, O.E. Karasan, B.Y. Kara, J.F. Campbell and A. Ozel, Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones. *Transp. Res. Part B: Methodol.* **168** (2023) 81–123.
[23] S.A. Vásquez, G. Angulo and M.A. Klapp, An exact solution method for the tsp with drone based on decomposition. *Comput. Oper. Res.* **127** (2021) 105127.
[24] P. Venkatesh, G. Srivastava and A. Singh, A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem, in Harmony Search and Nature Inspired Optimization Algorithms. Springer (2019) 279–292.
[25] X. Zang, L. Jiang, M. Ratli and B. Ding, A parallel variable neighborhood search for solving covering salesman problem. *Optim. Lett.* **16** (2022) 175–190.
[26] H. Zhang and Y. Xu, Online covering salesman problem. *J. Comb. Optim.* **35** (2018) 941–954.