BRANCH-AND-CUT ALGORITHMS FOR THE Covering Salesman Problem

A Preprint

Lucas Porto Maziero
Institute of Computing
University of Campinas
Av. Albert Einstein 1251, 13083-852, Campinas, SP, Brazil
lucas.maziero@ic.unicamp.br

Fábio Luiz Usberti
Institute of Computing
University of Campinas
Av. Albert Einstein 1251, 13083-852, Campinas, SP, Brazil
fusberti@ic.unicamp.br

Celso Cavallucci
Institute of Computing
University of Campinas
Av. Albert Einstein 1251, 13083-852, Campinas, SP, Brazil
celsocv@ic.unicamp.br

April 5, 2021

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.

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

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 [1] 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.
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 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, 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. [5], 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. [6] 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, González, and Toth [7] 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 [8] propose 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 $\alpha = \frac{1}{4} + \frac{4(k + 2)r}{OP T + 8} + 2\gamma \rho$, $\nu$ is the approximation ratio for the Steiner Tree Problem, $\rho$ 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 (Dumitrescu and Mitchell [9], Gulczynski et al. [10], Dong et al. [11], Shuttleworth et al. [12], Behdani and Smith [13], Coutinho et al. [14]).

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 [11], 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 [15] revisited the
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                  | ✓                              | ✓                      | ✓                  | ✓                      |

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 et al. [16] give 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.

More recently, Venkatesh et al. [17] proposed a multi-start iterated local search algorithm for the CSP and incorporated a variable degree of perturbation strategy to further improve the solution obtained through their heuristic approach. Computational results show that the proposed approach is competitive with other state-of-the-art heuristic approaches for solving the CSP. Zang et al. [13] reformulated the CSP as a bilevel CSP (BCSP) 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. Venkatesh et al. [19] developed two hybrid metaheuristic approaches for the CSP. The first is based on the artificial bee colony algorithm (ABC) and the second is based on the genetic algorithm (GA). Both approaches were competitive with the state-of-the-art of heuristic approaches for the CSP.

**Our contribution:** despite being well studied in the heuristic front, 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. To address this matter, the first branch-and-cut framework is proposed for the CSP. 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 compares 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, 1b and 1c show optimal solutions for three CSP instances with 200 vertices.

An integer linear programming (ILP) formulation for the CSP is presented. Binary variable \( x_e \) shows if an edge \( e \in E \) belongs (1) or not (0) to the tour and binary variable \( y_v \) shows 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 \( S \subseteq V \) and the other in \( V \setminus S \) and \( E(S) \) the set of edges with both endpoints in \( S \).
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.
\[(CSP) \quad \begin{align*} 
\text{MIN} & \quad \sum_{e \in E} c_e x_e, \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 2y_v \quad \forall v \in V, \\
& \quad \sum_{i \in C(v)} y_i \geq 1 \quad \forall v \in V, \\
& \quad \sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1) \quad \forall S \subset V, i \in S, j \in V \setminus S, \\
& \quad x_e \in \{0, 1\} \quad \forall i, j \in V, \\
& \quad y_v \in \{0, 1\} \quad \forall i \in V. 
\end{align*} \]

The CSP formulation is derived from the GTSP formulation proposed by Fischetti, González, and Tóth [7]. 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). Constraints (3) impose that each vertex must be covered at least once. Constraints (4) are subtour elimination constraints which state that every cut separating two vertices in the tour contains at least two edges.

### 3 Valid Inequalities

This section presents valid inequalities proposed by Fischetti, González, and Tóth [7] 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 \(C(u) = C(v)\).

![Figure 2: Example of \(S \notin \gamma(V)\) (a) and \(S \in \gamma(V)\) (b).](image)

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 of vertices that contains \(C(v)\) for at least one vertex \(v \in V\), i.e., \(\gamma(V) = \{F \subseteq P(V) : \forall S \in F, \exists v \in S, C(v) \subseteq S\}\)
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 2b, 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:

\[
\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \in \gamma(V) : D(S) \neq V, \quad (7)
\]

\[
\sum_{e \in \delta(S)} x_e \geq 2y_i \quad \forall S \notin \gamma(V) : D(S) \neq V, i \in S, \quad (8)
\]

\[
\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1) \quad \forall S \notin \gamma(V) : D(S) = V, i \in S, j \in V \setminus S. \quad (9)
\]

Inequalities (1) ensure that each cut separating two sets $C(v)$ and $C(u)$ 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), (8), and (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.

### 3.1 Cover Intersection Inequalities

Consider the case in which two covering sets $C(v)$ and $C(u)$, for some pair of vertices $v$ and $u$, overlap. This is a typical 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 $C(v) \cap C(u)$.

\[
\sum_{e \in (\delta(S) \cup \delta(S_v))} x_e \geq 2 \quad \forall v \in V, \forall S \subset V : S \in \gamma(V) \quad (10)
\]

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.

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 $C(u)$, 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 $\delta(S \setminus S_v)$. In Figure 3a, a feasible solution is presented in which both covering sets $C(u)$ and $C(v)$ are visited by the same tour. Despite the fact that the edge cut-set $\delta(C(u))$ is empty, it is not a concern, since $\delta(C(u) \cap C(v))$ contains two edges. This is not the case in Figure 3b, where both $\delta(C(u))$ and $\delta(C(u) \cap C(v))$ are empty, asserting the infeasibility. It is worth mentioning that the solution depicted in Figure 3b violates the CI inequality associated to the set $S = C(u)$ and vertex $v$. 

![Figure 3: Example of feasible (a) and infeasible (b) solutions in the context of overlap of covering sets.](image-url)
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 \( x^I, y^I \) and \( x^F, y^F \) 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 \( x^I, y^I \) and \( x^F, y^F \), respectively. In \( G^I \), every vertex \( v \in V^I \) has a weight \( y_v \), such that \( y_v \in y^I \), and every edge \( e \in E^I \) has a cost \( x_e \), such that \( x_e \in x^I \). Similarly, in \( G^F \), every vertex \( v \in V^F \) has a weight \( y_v \), such that \( y_v \in y^F \), and every edge \( e \in E^F \) has a cost \( x_e \), such that \( x_e \in x^F \).

4.1 Separation routine for integer solutions

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

Let \( S \subseteq V \) 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 \( S \subseteq \gamma(V) \). 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 \( S \not\subseteq \gamma(V) \). By taking \( S_{aug} = S \cup C(5) \), then \( S_{aug} \in \gamma(V) \), however \( S_{aug} \) would not generate an effective cut, since vertex \( v_7 \in V^I \). Otherwise, effective cuts can be derived from \( S_{aug} = S \cup C(v_4) \) or \( S_{aug} = S \cup C(v_6) \). Therefore, for an inequality (7) or (10) associated to \( S_{aug} \) to be effective in cutting solution \( x^I, y^I \), 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 to each subcycle found in \( x^I, y^I \). The overall complexity of Algorithm 1 is bounded by \( O(V^2) \).

![Diagram Figure 4: Example of an invalid CSP solution with two subcycles.](image)

4.2 Exact separation routine for fractional solutions

This section gives the exact separation routines of inequalities (7)-(10) for a fractional CSP solution \( x^F, y^F \). In particular, the separation of inequalities (7), (8) and (9) follows the methodology proposed by Fischetti, González and Toth [7] for the GTSP. As for the CI inequalities (10), a transformation of the solution graph \( G^I \) is proposed to tackle the overlap of covering sets. The routines are described next.

As observed by Fischetti, González and Toth [7], the separation problem to find one or more inequalities (9) violated by \( x^F, y^F \) 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 \( j \) [20]. Similarly, the separation of inequalities (10) can...
Algorithm 1 Separation routine for integer solutions.

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

1: for each subcycle $S$ in $\{x^I, y^I\}$ do
2: $T \leftarrow \emptyset$
3: if $D(S) \neq V$ then
4: if $S \in \gamma(V)$ then
5: $T \leftarrow T \cup$ inequality (7) associated to $S$
6: else
7: $T \leftarrow T \cup$ inequality (8) associated to $S$
8: for each $v \in S$ do
9: $S_{aug} \leftarrow S \cup C(v)$
10: if $S_{aug} \cap (V^I \setminus S) = \emptyset$ then
11: if $D(S_{aug}) \neq V$ then
12: $T \leftarrow T \cup$ CI inequality (10) associated to $S_{aug}$
13: else
14: for $u \in V$ do
15: $S_u \leftarrow S \cap C(u)$;
16: if $\delta(S_u) \cap E^I = \emptyset$ then
17: $T \leftarrow T \cup$ CI inequality (10) associated to $S_{aug}$ and $S_u$
18: else
19: for each subcycle $S'$ in $\{x^I, y^I\} : S' \neq S$ do
20: $T \leftarrow T \cup$ inequality (9) associated to $S$ and $S'$
21: return $T$;

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 $t$ [20], 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. 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 6.

An exact separation algorithm for CI inequalities (10) is proposed to accommodate 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 $\sum_{e \in \delta(S_{min})} x_e$ be the minimum cut between $C(v)$ and $C(u)$ and $S_u = S_{min} \cap C(u)$. If $\sum_{e \in \delta(S_{min}) \cup \delta(S_u)} x_e$ 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 [20] 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 $\epsilon$ 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) and (10).
Figure 5: Max-flow instance for the separation of inequality (7) in the case where $C(v) \cap C(u) = \emptyset$.

(a) Pre-augmentation

(b) Augmented graph

Figure 6: Graph augmentation for the separation of CI inequalities (10).
Algorithm 3 Exact separation routine for fractional solutions.

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

1: \( T \leftarrow \emptyset \)
2: \textbf{for} \( v \in V \) \textbf{do}
3: \hspace{1em} \textbf{for} \( u \in V \setminus \{v\} \) \textbf{do}
4: \hspace{2em} \textbf{if} \( D(C(v)) \neq V \) and $C(v) \cap C(u) = \emptyset$ \textbf{then}
5: \hspace{3em} $S \leftarrow \text{minCut}(C(v), C(u), G^F)$
6: \hspace{3em} $T \leftarrow T \cup \text{inequality (7)}$ associated to $S$.
7: \hspace{2em} \textbf{else}
8: \hspace{3em} $G^F_{\text{aug}} \leftarrow \text{augment}(G^F)$
9: \hspace{3em} $S \leftarrow \text{minCut}(C(v), C(u), G^F_{\text{aug}})$
10: \hspace{3em} $T \leftarrow T \cup \text{CI inequality (10)}$ associated to $S$ and $u$.
11: \hspace{2em} \textbf{if} \( y_v > 0 \) and $v \notin C(u)$ \textbf{then}
12: \hspace{3em} $S \leftarrow \text{minCut}(v, C(u), G^F)$
13: \hspace{3em} $T \leftarrow T \cup \text{inequality (8)}$ associated to $S$ and $u$.
14: \hspace{2em} \textbf{if} \( y_v + y_u - 1 > 0 \) \textbf{then}
15: \hspace{3em} $S \leftarrow \text{minCut}(v, u, G^F)$
16: \hspace{3em} $T \leftarrow T \cup \text{inequality (9)}$ associated to $S$, $v$ and $u$.
17: \textbf{return} $T$;

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

4.3 Heuristic separation routine for fractional solutions

A heuristic separation has the purpose of finding inequalities being violated by a fractional solution $\{x^F, y^F\}$ 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 to $S$ cuts $\{x^F, y^F\}$; (ii) if $D(S) = V$ and $\sum_{e \in \delta(S)} x_e + \sum_{e \in \delta(S_u) \setminus \delta(S)} x_e < 2$ for some vertex $v \in V \setminus \{u\}$, then the CI inequality (10) associated to $S$ and $v$ cuts $\{x^F, y^F\}$.

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

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

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

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.
Algorithm 3 Heuristic separation routine for fractional solutions.

Input: graph $G^F(V^F, E^F)$ induced by a fractional solution $\{x^F, y^F\}$ for the CSP.

Output: a set $T$ of valid inequalities that cuts $\{x^F, y^F\}$.

1: $T \leftarrow \emptyset$
2: for $u \in V$ do
3: $S \leftarrow C(u)$;
4: if $D(S) \neq V$ then
5: if $\sum_{e \in \delta(S)} x_e < 2$ then
6: $T \leftarrow T \cup$ inequality (7) associated to $S$.
7: else
8: for $v \in V : v \neq u$ do
9: $S_v \leftarrow S \cap C(v)$;
10: if $\sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ then
11: $T \leftarrow T \cup$ CI inequality (10) associated to $S$ and $v$.
12: Compute the connected components $S_1, \ldots, S_p$ of $G^F$;
13: for $k = 1, \ldots, p$ do
14: $S \leftarrow S_k$
15: if $S \in \gamma(V)$ then
16: if $D(S) \neq V$ then
17: $T \leftarrow T \cup$ inequality (7) associated to $S$.
18: else
19: for $v \in V : v \neq w$ do
20: $S_v \leftarrow S \cap C(v)$;
21: if $\sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ then
22: $T \leftarrow T \cup$ CI inequality (10) associated to $S$ and $v$.
23: else
24: $i \leftarrow \arg \max \{y_v : v \in S\}$.
25: if $D(S) \neq V$ then
26: $T \leftarrow T \cup$ inequality (8) associated to $S$ and $i$.
27: for $l = k, \ldots, p$ do
28: $j \leftarrow \arg \max \{y_v : v \in S_l\}$
29: $T \leftarrow T \cup$ inequality (9) associated to $S$, $i$ and $j$.
30: return $T$

5.1 Instances

The set of instances used in the computational experiments was created by Salari et al.\[16\] based on the TSPLIB \[21\], and they are divided in two types: small (36 instances) and medium (12 instances). The small instances have $51 \leq |V| \leq 100$ vertices and medium-size instances have $150 \leq |V| \leq 200$ vertices. There is also a set of large instances with $|V| \geq 532$, which Salari et al. assigned only for the testing of heuristics, and for this reason, the large instances are not reported in this paper. The covering set of each vertex is defined by its $k$ closest vertices. For each graph, three values of $k$ were used, $k = 7, k = 9$, and $k = 11$. Full experimental data (including results for large instances), instances, and source codes are available on-line.\[4\]

5.2 Computational Settings

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

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), (8), and (9), but excluding the CI inequalities (10).

\[http://www.ic.unicamp.br/~fusberti/problems/csp\]
2. CSP-I&Fvp: on the root node, exact separation routine for fractional solutions (Algorithm 2) considering inequalities (7), (8), and (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 Section 4.2).

3. CSP-I&Fvp-X: same as CSP-I&Fvp, but including the CI inequalities (10);

4. CSP-I&Fh: on the root node, exact separation routine for fractional solutions (Algorithm 2) considering inequalities (7), (8), and (9), but excluding the CI inequalities (10). For the non-root nodes, heuristic separation for fractional solutions (Algorithm 3) considering inequalities (7), (8), and (9), but excluding the CI inequalities (10);

5. CSP-I&Fh-X: same as CSP-I&Fh, but including inequalities (10);

These methodologies were compared with the integer linear programming formulation proposed by Salari et al.\cite{16}, 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&Fh and CSP-I&Fh-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 and medium instances in Tables 2 and 3 respectively. Each table reports for each methodology and for each instance, the following:

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

In both tables, 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, the Tables 2 and 3 highlight the optimal solutions (underlined) and the best lower bounds (in bold) obtained by each methodology.
Table 2: Results of computational experiments for the small-size instances.

| Instance | BestUB | References | LB | Gap | Time |
|----------|--------|------------|-----|-----|------|
| eil51    | 164    | 149        | 164 | 4   | 2    |
| 9        | 159    | 220        | 159 | 2   | 2    |
| 11       | 147    | 681        | 147 | 2   | 2    |
| berlin52 | 3887   | 140        | 3887 | 100 | 2    |
| 9        | 3340   | 212        | 3340 | 100 | 2    |
| 11       | 3262   | 255        | 3262 | 100 | 2    |
| st70     | 288    | 7          | 288 | 7   | 5    |
| 9        | 259    | 3          | 259 | 7   | 6    |
| 11       | 247    | 13.14      | 247 | 7   | 14   |
| eil76    | 207    | 6          | 207 | 9   | 15   |
| 9        | 185    | 10         | 185 | 9   | 13   |
| 11       | 170    | 8          | 170 | 7   | 15   |
| pr76     | 50275  | 7          | 50275 | 100 | 7    |
| 9        | 45348  | 6          | 45348 | 100 | 10   |
| 11       | 43028  | 7          | 43028 | 100 | 14   |
| rat99    | 486    | 19         | 486 | 17  | 33   |
| 9        | 455    | 30         | 455 | 27  | 28   |
| 11       | 444    | 32         | 444 | 138 | 37   |

Continued on next page
Continued from previous page

| Instance | BestUB | SRS | CSP | CSP-I | CSP-I&Fvp | CSP-I&Fh | CSP-I&Fvp-X | CSP-I&Fh-X |
|----------|--------|-----|-----|-------|----------|---------|------------|------------|
|          |        |     |     | LB    | Gap      | Time    | LB         | Gap        | Time       | LB         | Gap        | Time       | LB         | Gap        | Time       |
| kroA100  |    7   | 7674| 9177| 9074  | 0        | 15      | 9074       | 0          | 18        | 9074       | 0          | 27        | 9074       | 0          | 30        |
| 9        | 9159   | 7674| 13.38| 3600  | -41       | 1800    | 9159       | 0          | 8400      | 0          | 7744       | 0          | 18400      | 0          | 20900     |
| 11       | 8901   | 7674| 3.59 | 3600  | 10324     | 24600   | 8901       | 0          | 45        | 8901       | 0          | 45        | 8901       | 0          | 79        |
| kroB100  |    7   | 9537| 9674| 9537  | 0        | 45      | 9537       | 0          | 22        | 9537       | 0          | 26        | 9537       | 0          | 23        |
| 9        | 9240   | 9674| 20.34| 3600  | -2040     | 2040    | 9240       | 0          | 21        | 9240       | 0          | 21        | 9240       | 0          | 28        |
| 11       | 8842   | 9674| -   | 3600  | -3600     | 9674    | 8842       | 0          | 25        | 8842       | 0          | 29        | 8842       | 0          | 36        |
| kroC100  |    7   | 9723| 9723| 9723  | 0        | 107     | 9723       | 0          | 102       | 9723       | 0          | 67        | 9723       | 0          | 92        |
| 9        | 9171   | 9723| 19.68| 3600  | -217      | 217     | 9171       | 0          | 45        | 9171       | 0          | 783       | 9171       | 0          | 972       |
| 11       | 8632   | 9723| 13.73| 3600  | -254      | 254     | 8632       | 0          | 38        | 8632       | 0          | 820       | 8632       | 0          | 870       |
| kroD100  |    7   | 9626| 9626| 9626  | 0        | 17      | 9626       | 0          | 20        | 9626       | 0          | 34        | 9626       | 0          | 23        |
| 9        | 8985   | 9626| -   | 3600  | -3600     | 9626    | 8985       | 0          | 22        | 8985       | 0          | 27        | 8985       | 0          | 35        |
| 11       | 8725   | 9626| -   | 3600  | 1980      | 1980    | 8725       | 0          | 35        | 8725       | 0          | 48        | 8725       | 0          | 80        |
| kroE100  |    7   | 10150| 10150| 10150 | 0        | 81      | 10150      | 0          | 42        | 10150      | 0          | 76        | 10150      | 0          | 32        |
| 9        | 8991   | 10150| 7.77 | 3600  | -336      | 336     | 8991       | 0          | 31        | 8991       | 0          | 55        | 8991       | 0          | 88        |
| 11       | 8450   | 10150| 9.19 | 3600  | -237      | 237     | 8450       | 0          | 23        | 8450       | 0          | 261       | 8450       | 0          | 193       |
| rd100    |    7   | 3461| 3461| 3461  | 0        | 119     | 3461       | 0          | 20        | 3461       | 0          | 20        | 3461       | 0          | 22        |
| 9        | 3194   | 3461| 19.90| 3600  | -18       | 18      | 3194       | 0          | 18        | 3194       | 0          | 18        | 3194       | 0          | 25        |
| 11       | 2922   | 3461| 10.33| 3600  | -28       | 28      | 2922       | 0          | 21        | 2922       | 0          | 20        | 2922       | 0          | 27        |
| Avg      |        |     |      |       |          |        |           |            |           |            |            |           |            |             |             |            |
Table 3: Results of computational experiments for the medium-size instances.

| Instance | NC | Best UB | Reference(s) | Best UB | Reference(s) | Best UB | Reference(s) | Best UB | Reference(s) | Best UB | Reference(s) |
|----------|----|---------|---------------|---------|---------------|---------|---------------|---------|---------------|---------|---------------|
| kroA150  | 7  | 11423   | [6, 15, 16, 17] | 11423   | [6, 15, 16, 17] | 11423   | [6, 15, 16, 17] | 11423   | [6, 15, 16, 17] | 11423   | [6, 15, 16, 17] |
| kroB150  | 7  | 11457   | [6, 15, 16, 17] | 11457   | [6, 15, 16, 17] | 11457   | [6, 15, 16, 17] | 11457   | [6, 15, 16, 17] | 11457   | [6, 15, 16, 17] |
| kroA200  | 7  | 13285   | [6, 15] | 11660 | [6, 15] | 12611 | [6, 15] | 13108 | [6, 15] | 1.35 | 3600 |
| kroB200  | 7  | 13051   | [15, 16, 17] | 12262 | [15, 16, 17] | 12904 | [15, 16, 17] | 13051 | [15, 16, 17] | 1.35 | 3600 |
| Avg      | 10462.33 | 6.12 | 3897.42 | 10886.58 | 1.98 | 1718.00 | 11077.50 | 0.31 | 1220.67 | 11010.50 | 0.86 | 1681.92 | 11102.42 | 0.11 | 808.42 |
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&Fvp and CSP-I&Fvp-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, 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&Fh-X, with an average gap of 0.11%. Now, the heuristic separation overcomes the exact separation, mainly due to the reduction in the computational effort. The worst-performing methodology (CSP-I) obtained an average gap of 6.12%, showing that by using only integral cuts performs poorly for more challenging instances.

The effect of the CI inequalities (10) in the performance of the methodologies was also examined. Comparing CSP-I&Fvp and CSP-I&Fvp-X, their average gaps were both zero for small instances and reduced from 1.98% to 0.86% for medium instances. Moreover, comparing CSP-I&Fh and CSP-I&Fh-X, their average gaps reduced from 0.08% to 0.03% for small instances and reduced from 0.31% to 0.11% for medium instances. Therefore, the CI inequalities are confirmed to have a significant impact on reducing the optimality gaps.

Previously, from 48 small and medium-size CSP instances, only 9 optimal solutions were known. These computational results have shown that the 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 38 instances were proven optimal for the first time.

6 Final Remarks

The proposed branch-and-cut framework for the CSP uses existing valid inequalities for the GTSP, by Fischetti et al. [7], 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 delves into the effectiveness of the framework. The overall results show unequivocally the branch-and-cut methodologies outperforming the best known exact methodology from literature and unveiling 38 new optimal solutions. The experiments also show that the CI inequalities had a major role in the performance of the methodologies.

The ideas presented in this work can support the exact solution of many possible developments 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.
Acknowledgments

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

References

[1] John R Current and David A Schilling. The covering salesman problem. Transportation science, 23(3):208–213, 1989.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton, NJ, USA, 2007.

[3] John Current, Hasan Pirkul, and Erik Rolland. Efficient algorithms for solving the shortest covering path problem. Transportation Science, 28(4):317–327, 1994.

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

[5] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The covering tour problem. Operations Research, 45(4):568–576, 1997.

[6] Bruce Golden, Zahra Naji-Azimi, S Raghavan, Majid Salari, and Paolo Toth. The generalized covering salesman problem. INFORMS Journal on Computing, 24(4):534–553, 2012.

[7] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research, 45(3):378–394, 1997.

[8] Huili Zhang and Yinfeng Xu. Online covering salesman problem. Journal of Combinatorial Optimization, pages 1–14, 2018.

[9] Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. Journal of Algorithms, 48(1):135–159, 2003.

[10] Damon J Gulczynski, Jeffrey W Heath, and Carter C Price. The close enough traveling salesman problem: A discussion of several heuristics. In Perspectives in Operations Research, pages 271–283. Springer, 2006.

[11] Jing Dong, Ning Yang, and Ming 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, pages 145–163. Springer, 2007.

[12] Robert Shuttleworth, Bruce L Golden, Susan Smith, and Edward W asil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In The Vehicle Routing Problem: Latest Advances and New Challenges, pages 487–501. Springer, 2008.

[13] Behnam Behdani and J Cole Smith. An integer-programming-based approach to the close-enough traveling salesman problem. INFORMS Journal on Computing, 26(3):415–432, 2014.

[14] Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. INFORMS Journal on Computing, 28(4):752–765, 2016.

[15] Majid Salari and Zahra Naji-Azimi. An integer programming-based local search for the covering salesman problem. Computers & Operations Research, 39(11):2594–2602, 2012.

[16] Majid Salari, Mohammad Reihaneh, and Mohammad S Sabbagh. Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. Computers & Industrial Engineering, 83:244–251, 2015.
[17] Pandiri Venkatesh, Gaurav Srivastava, and Alok 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*, pages 279–292. Springer, 2019.

[18] Xiaoning Zang, Li Jiang, Mustapha Ratli, and Bin Ding. A parallel variable neighborhood search for solving covering salesman problem. *Optimization Letters*, pages 1–16, 2020.

[19] Venkatesh Pandiri, Alok Singh, and André Rossi. Two hybrid metaheuristic approaches for the covering salesman problem. *Neural Computing and Applications*, pages 1–21, 2020.

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

[21] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.