Exact approach for the 2-Vertex-Connected Star problem

Louis Kurdyk
louis.kurdyk@dauphine.psl.eu
Université Paris Dauphine - PSL,
CNRS, LAMSADE UMR 7243
Paris, France

André Rossi
andre.rossi@dauphine.psl.eu
Université Paris Dauphine - PSL,
CNRS, LAMSADE UMR 7243
Paris, France

Sonia Toubaline
sonia.toubaline@dauphine.psl.eu
Université Paris Dauphine - PSL,
CNRS, LAMSADE UMR 7243
Paris, France

Abstract

This paper considers the 2-Vertex-Connected Star (2-VC-S) problem, motivated by telecommunications network design and the survivability requirements arising in real-world applications. Given a complete graph with both edges and arcs, the goal is to partition the vertices into *hubs* and *terminals* to construct a minimum-cost subgraph observing two main characteristics. First, the subgraph induced by the vertices labelled as hubs and referred to as the *backbone* network, must use only edges and be 2-vertex-connected. Second, each terminal must be connected to a hub of the backbone network by an arc, forming the *tributary* network.

In order to solve instances to optimality and to improve upon some of the best known solutions for the problem, we propose an integer linear programming formulation solved using a branch-and-cut approach. The computational results indicate that the 2-VC-S problem, that permits more general backbones than the Ring–Star problem, sometimes leads to practical cost reductions.

Keywords

2-vertex-connectivity, fault-tolerant network design, ear decomposition, integer linear programming

1 Introduction

In telecommunications network design, traffic is often aggregated and routed through multiple network layers. When considering two layers, the network is typically structured around a set of hubs responsible for traffic aggregation, while terminals connect to these hubs to inject and retrieve traffic. The network induced by the hubs, referred to as the *backbone* network, ensures communication between terminals through the hub layer, while the *tributary* network connects each terminal to a hub.

While service availability is often a primordial aspect of network design, as networks scale in size, complexity, and number of layers, resilience to failures becomes an important aspect. Survivable network design has been extensively studied in the literature, mostly in the context of single-level networks where all vertices are treated uniformly and connectivity requirements are imposed on the entire graph [3]. In hub-based architectures, however, survivability concerns primarily the backbone network, whose failure may disconnect a large number of terminals.

Using a tree-shaped backbone induces vulnerability, since any single hub or link failure disconnects the network. In contrast, a cycle-shaped backbone provides alternate routing paths and ensures survivability under a single failure. Such structures are therefore needed in applications including the Digital Data Service problem studied by Xu *et al.* [11] or the embedding of fault-tolerant logical networks over survivable physical networks considered by Zetani *et al.* in [12]. However, Monma showed in [1] that cycles are not necessarily the cheapest way to build a 2-vertex-connected subgraph. With this in mind, Recoba *et al.* [7] introduced the following topological generalization of the Ring-Star Problem (RSP) solved by Labbé in [4].

The 2-Vertex-Connected Star (2-VC-S) Problem can be defined as follows. Given a complete graph $G = (V, E \cup A)$ using both arcs and edges with two associated cost functions, $c : E \to \mathbb{R}$ for inter-hub edges and $d : A \to \mathbb{R}$ for connecting a terminal to a hub, the goal is to build a minimum-cost subgraph $G' = (V, E' \cup A')$ such that the vertex set V is partitioned into a set H of hubs and a set T of terminals. Each terminal in T must be connected by an arc in A' to a hub in H, and the hubs must form a 2-vertex-connected subgraph using edges from E'. In [7], it is proven that the problem is NP-complete and a Greedy Randomized Adaptive Search Procedure (GRASP) is provided to compute heuristic solutions. While these solutions often come close in term of objective value to those obtained by solving the RSP on the same instances, they struggle to significantly reduce the cost of the resulting networks. In this regard, we aim to provide an exact model based on an integer linear programming formulation to either identify improvements permitted by the generalized backbone topology or certify that the currently known solutions are actually minimum-cost 2-vertex-connected star networks for the corresponding instances.

The paper is organized as follows. Section 2 presents the integer linear programming (ILP) model used within the branch-and-cut framework and details how 2-vertex-connectivity is imposed on the backbone network. Section 3 describes the solution process, including a variation of the heuristic proposed by Recoba *et al.*, a primal heuristic based on open-ear decompositions, and the separation procedures employed. Finally, numerical experiments are presented in Section 4, in order to assess the impact of allowing generalized 2-vertex-connected backbone topologies.

2 ILP model for the 2-VC-S

To solve the general setting of the 2-VC-S problem, we propose an ILP model based on two properties. Those properties are introduced alongside the used decision variables in Section 2.1 before being leveraged in the full ILP model presented in Section 2.2.

2.1 Enforcing the backbone connectivity and 2-vertex-connectivity

The ILP formulation for the 2-VC-S problem uses the following binary decision variables:

- $h_i = 1$ if vertex i is a hub, for all i in V, 0 if it is a terminal.
- $x_{ij} = 1$ if edge (i, j) connects two hubs in the backbone, for all (i, j) in E, 0 otherwise.
- $y_{ij} = 1$ if arc (i, j) is used to connect terminal i to hub j, for all $(i, j) \in A$, 0 otherwise.

In order to enforce connectivity, we show that the following integer nonnegative quantity is zero if and only if the nonempty

INOC '26, Liège (Belgium)

10.48786/inoc.2026.09

set $S \subseteq V$ is a connected component of $G[H]$, where H is any set of hubs, and $G[H]$ is the subgraph of G induced by H:

$$\left(|S| - \sum_{i \in S} h_i\right) + \sum_{e \in \delta(S)} x_e$$

with for a given set $S \subseteq V$, we use the notation $\delta(S) = \{(i,j) \in E \mid i \in S \oplus j \in S\}$ with $\oplus$ being the XOR logical operator.

First, the quantity in parentheses is a nonnegative integer, and is zero if and only if all the elements of S are hubs. The second term is also a nonnegative integer, that is zero if and only if no vertex in S has a neighbor in $V \backslash S$. It follows that this quantity is a nonnegative integer, that is zero if and only if S is a connected component of $G[H]$. In the context of 2-VC-S, we can state the constraint:

$$|S| - \sum_{i \in S} h_i + \sum_{e \in \delta(S)} x_e \geq 2h_j \quad \forall S \subset V, |S| \geq 2, \forall j \in V \backslash S \quad (1)$$

This constraint enforces that $G[H]$ is 2-edge-connected, which is a necessary condition for 2-vertex-connectivity. If S contains H, there is no hub $j \in V \setminus S$, so the constraint is trivially satisfied. Otherwise, if $|S| \geq 2$ and S is not a connected component which is maximal, the left-hand side (LHS) is strictly positive and cannot be less than 2, since the minimum degree of any vertex in a 2-vertex-connected graph is 2. Therefore, the LHS is always at least $2h_j$ for all $j \in V \setminus S$. For singleton sets S, the 2-edge-connectivity requirement is ensured by constraint (5), which enforces that each hub has degree at least 2.

Similarly, to enforce 2-vertex-connectivity, we show that for all $u \in V$, the following integer nonnegative quantity is zero if and only if the nonempty set $S \subseteq V \backslash \{u\}$ is a connected component of $G[H \backslash \{u\}]$:

$$\left(|S| - \sum_{i \in S} h_i\right) + \sum_{\substack{(i,j) \in \delta(S) \\ u \neq i,j}} x_{ij}$$

The quantity in parentheses is a nonnegative integer, and is zero if and only if all the elements of S are hubs. The second term is also a nonnegative integer, that is zero if and only if no vertex in S has a neighbor in $V \backslash (S \cup \{u\})$. It follows that this quantity is a nonnegative integer, that is zero if and only if S is a connected component of $G[H \backslash \{u\}]$. Consequently, we enforce the following constraint:

$$|S| - \sum_{i \in S} h_i + \sum_{\substack{(i,j) \in \delta(S) \\ u \neq i,j}} x_{ij} \geq h_j \quad \begin{array}{l} \forall u \in V, \\ \forall S \subset V \setminus \{u\},\ |S| \geq 1, \\ \forall j \in V \setminus (S \cup \{u\}) \end{array} \quad (2)$$

This constraint enforces that removing the vertex u from the backbone $G[H]$ leaves a connected subgraph, implying the 2-vertex-connectivity of $G[H]$. Indeed, the constraint is satisfied if S contains $H \backslash \{u\}$ because there is no hub j in $V \backslash (S \cup \{u\})$. If S is not a connected component which is maximal of $G[H \backslash \{u\}]$, the inequality is satisfied because its LHS is strictly positive, so it exceeds h_j for all j in $V \backslash (S \cup \{u\})$.

2.2 The ILP model

The ILP model to solve the 2-VC-S problem is given by:

$$\begin{aligned}
\min \quad & \sum_{(i,j) \in E} c_{ij} x_{ij} + \sum_{(i,j) \in A} d_{ij} y_{ij} + \sum_{i \in V} f_i h_i \\
\text{s.t.} \quad & (1) - (2) \\
& \sum_{i \in V} h_i \geq 3 && (3) \\
& \sum_{\substack{j \in V \\ i \neq j}} y_{ij} = 1 - h_i && \forall i \in V \quad (4) \\
& \sum_{\substack{j \in V \\ i \neq j}} x_{ij} \geq 2h_i && \forall i \in V \quad (5) \\
& x_{ij} + y_{ij} \leq h_j && \forall (i,j) \in E \quad (6) \\
& x_{ij} + y_{ji} \leq h_i && \forall (i,j) \in E \quad (7) \\
& h_i \in \{0,1\} && \forall i \in V \quad (8) \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in E \quad (9) \\
& y_{ij} \in \{0,1\} && \forall (i,j) \in A \quad (10)
\end{aligned}$$

While the integrality constraint (10) on variables y_{ij} can be relaxed it can be checked that these variables take integer values whenever the other variables are integer ; it makes no practical difference in the resolution procedure. Constraint (3) ensures that the backbone network contains at least three hubs, since any 2-vertex-connected graph must have at least three vertices. Constraint (4) is there to ensure that each vertex is either used as a hub or is a terminal connected to a hub. As for constraint (5), it gives a lower bound on the degree of each hub of the backbone network, this minimum degree is set to 2 since if a hub had only one neighbor, it would become an articulation point. Constraints (6) and (7) impose that an edge (i,j) can only be used if both i and j are hubs and that an arc can only be directed toward a hub. Moreover, they ensure that either the edge (i,j) or the arc (i,j) can be present in a solution but not both.

3 Branch-and-cut framework

In order to solve the presented ILP, several procedures are implemented alongside the branch-and-cut algorithm. These approaches either rely on heuristics to improve the current upper bound or on the addition of cuts, focusing on strengthening the lower bound. They are described in the following subsections.

3.1 Initial solutions via a GRASP heuristic

Since the 2-VC-S problem and its NP-completeness are established by Recoba *et al.* [7], we adapt their GRASP heuristic to generate initial feasible solutions. Used as a warm start within the branch-and-cut framework, these solutions lead to a more efficient overall solution process, as briefly discussed in Section 4.2. For a complete description of the heuristic, we refer the reader to Recoba *et al.* [7], and only summarize the aspects relevant to our modifications.

The GRASP consists of a construction phase followed by a local search phase. The construction phase builds an initial feasible solution from a cycle; we use it exactly as described in [7] and do not modify it further.

The local search phase iteratively improves the current solution by applying six neighborhood operators. Two of these operators rely on exact ILP models. Although these ILP-based neighborhoods have proven effective in improving the objective value, we do not employ them in our implementation. While exact models can yield high-quality heuristic solutions given

sufficient time, our goal of producing good solutions fast does not justify this additional computational effort. Instead, we replace these operators with a decomposition procedure followed by a recursive call to the GRASP. An example of the resulting decomposition is presented below.

Example 3.1. Consider a possible backbone network over the set $\{1, \ldots, 10\}$ of hubs as shown in Figure 1.

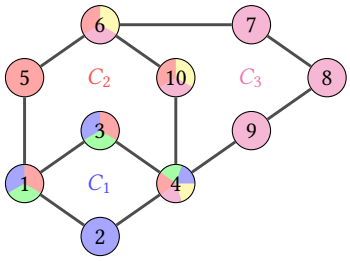


Figure 1: Decomposition of a backbone network into cycles and paths

To decompose the graph in Figure 1 into the three cycles $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{1, 3, 4, 10, 6, 5\}$, and $C_3 = \{4, 9, 8, 7, 6, 10\}$, and two paths $P_1 = \{1, 3, 4\}$ and $P_2 = \{6, 10, 4\}$, we proceed in three main steps.

First, we apply a chain decomposition as described in [9], which produces chains ordered such that the first chain is the only cycle and the rest are paths.

Next, we reorganize the chains to reduce the size of the longest one. For example, if the initial decomposition gives $\Gamma_1 = \{1, 2, 3, 4\}$, $\Gamma_2 = \{1, 5, 6, 7, 8, 9, 4\}$, and $\Gamma_3 = \{6, 10, 4\}$, we compute a shortest path from the endpoints of Γ_2, 1 and 4, using edges from Γ_2 and Γ_3. This produces a new chain $\Gamma_2' = \{1, 5, 6, 10, 4\}$, and the unused edges form $\Gamma_3' = \{6, 7, 8, 9, 4\}$, which respectively replace Γ_2 and Γ_3. This prevents the creation of excessively large cycles, which would increase the computational cost of recursively applying the heuristic.

Finally, to obtain the final cycles and paths, we iteratively compute shortest paths between the endpoints of each chain starting from the second one, closing off the cycles and producing the paths.

This decomposition allows us to locally optimize each cycle independently. Since the GRASP heuristic is designed to construct 2-vertex-connected star subgraphs, it can be applied to each cycle. Doing so may yield cheaper structures than simple cycles on the same vertex sets, or it may introduce terminals to reduce the overall cost.

However, vertices belonging to multiple components must remain hubs. Removing them from the backbone would break the 2-vertex-connectivity of the overall structure. Therefore, during each recursive call, we impose that such hubs cannot be reassigned as terminals. Under this restriction, each cycle can be optimized independently without risking a loss of global connectivity. If an improvement is obtained for multiple cycles, only the cycle providing the best improvement is replaced by the new 2-vertex-connected star subgraph returned by the recursive GRASP call.

We apply a similar strategy to locally optimize paths, enforcing the same constraints on hubs present in multiple components. Since paths are 1-vertex-connected subgraphs in the original backbone, we adapt the recursive algorithm to search for path-star topologies instead of more expensive 2-vertex-connected star graphs.

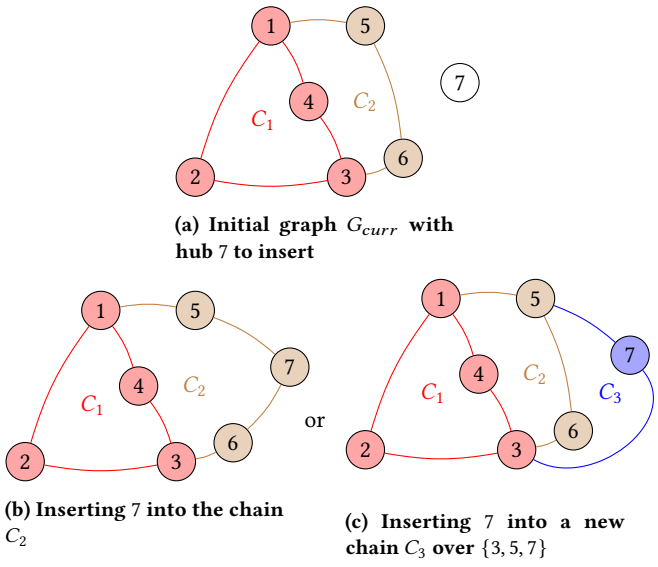
3.2 Rounding heuristic based on open-ear decomposition

In order to improve the performance of a branch-and-cut algorithm for solving an ILP model, fast primal heuristics are often employed to find integer solutions from the LP relaxation of the current node. To develop such an approach, we rely on the chain decomposition presented by Schmidt in [9] to iteratively build a 2-vertex-connected backbone network over a given set of hubs H derived from the LP relaxation. This strategy is described in Algorithm 1.

Algorithm 1: Rounding by open-ear decomposition

Input: h^*, x^* *the current fractional solution,* β *a rounding threshold*
Output: *A feasible solution to the 2-VC-S problem*
begin
  /* Define a cost function based on the LP */
  $c'_{ij} \leftarrow (1 - x^*_{ij})c_{ij} \quad \forall (i,j) \in E$
  /* Hub selection */
  $H \leftarrow \{v \in V \mid h_v \geq \beta\}$
  /* Initial chain: the only cycle */
  Select $\{u, v, w\} \subseteq H$ maximizing $c'_{u,v} + c'_{v,w} + c'_{u,w}$
  $E_{\text{chains}} \leftarrow \{\{(u,v),(v,w),(u,w)\}\}$
  $V_{\text{conn}} \leftarrow \{u, v, w\}$
  /* Add hubs to existing chains or construct new ones */
  while $V_{conn} \neq H$ **do**
    Pick $v \in H \setminus V_{\text{conn}}$ uniformly at random
    Find minimum cost option, with respect to c', to either:
    - Append v to an existing chain by replacing an edge (i,j) of the chain by (i,v) and (v,j), or
    - Create a new chain connecting v to V_{conn}
    Update E_{chains} accordingly
    $V_{\text{conn}} \leftarrow V_{\text{conn}} \cup \{v\}$
  /* Connect each terminal to its closest hub */
  $A' \leftarrow \emptyset$
  foreach $t \in V \setminus H$ **do**
    $A' \leftarrow A' \cup \{(t, \arg\min_{h \in H} d_{th})\}$
  return $(H, E_{\text{chains}}, A')$

The correctness of the algorithm follows from the observation that the constructed chains form an open-ear decomposition of a graph. Recall that an ear decomposition of a graph G is a partition of its edges into cycles and paths, called chains or ears, such that for each chain except the first one, its endpoints lie in previous ears while its internal vertices do not. This decomposition is called *open* if no chain except the first one is a cycle. As shown by Whitney [10], the existence of an open-ear decomposition implies that the graph is 2-vertex-connected, ensuring a feasible backbone network. We provide the following example to illustrate the algorithm.



(a) Initial graph G_{curr} with hub 7 to insert

(b) Inserting 7 into the chain C_2

(c) Inserting 7 into a new chain C_3 over $\{3, 5, 7\}$

Figure 2: Insertion procedure used by Algorithm 1

Example 3.2. Suppose that at a given iteration of Algorithm 1, the graph $G_{curr} = (V_{conn}, E_{chain})$ is given by Figure 2a. Then vertex 7 is a hub that remains to be inserted into the solution built. To do so, the algorithm offers two possibilities, either we split the edge of the chain C_1 or C_2 leading to the lowest cost increase (represented by Figure 2b) or we create a new chain C_3 from two distinct vertices (see Figure 2c). In any case, since we do not create any chain whose endpoints are identical except for C_1, we obtain a graph that is an open-ear decomposition.

When a new hub is inserted, the algorithm can either create a new chain or modify an existing one. Creating a new chain introduces two additional edges and removes none. By contrast, modifying a chain replaces an edge by two edges, adding only one edge overall. Since all modified costs c' are nonnegative, operations that introduce fewer new edges tend to limit the increase in total cost.

Initializing with a maximum-cost cycle makes expensive edges more attractive candidates for replacement, which promotes extending existing chains rather than creating new ones. In contrast, starting from a minimum-cost cycle discourages replacing its low-cost edges, leading to the creation of more new chains and typically higher overall cost.

3.3 Connectivity constraints separation

As constraints (1) and (2) are present in exponential number, a separation procedure is required in order to implement them efficiently. At integer nodes, separating violated inequalities (1) can be done by running a BFS over the backbone $B^* = (H^*, E^*)$ where $H^* = \{v \in V \mid h_v = 1\}$ and $E^* = \{e \in E \mid x_e = 1\}$. If the BFS does not reach every hub of the current solution then the reach set S can be associated a violated inequality (1), to be enforced. A similar approach can be derived to separate inequalities (2) by removing a vertex $u \in H^*$ and following the same procedure.

In order to improve performance, we strengthen the mathematical formulation by using the following constraints presented by Labbé *et al.* [4] adapted from the work of Lucena on Steiner problems [6].

$$\sum_{e \in \delta(S)} x_e \geq 2(h_i + h_j - 1) \quad \forall S \subsetneq V, \forall i \in S, \forall j \in V \setminus S \tag{11}$$

The intuition behind this constraint is that, in a 2-vertex-connected backbone network, we should be able to route two units of flow between any two hubs using edges with capacity 1. To detect violations of this constraint at fractional LP solutions, we compute a maximum flow at fractional nodes considering the vectors $\hat{h}$ and $\hat{x}$, which are the optimal LP relaxation solution at the current node.

To do so, we define an auxiliary capacitated graph with vertices $\hat{H} = \{v \in V \mid \hat{h}_v > 0\}$ and edges $\hat{E} = \{(i, j) \in E \mid i, j \in \hat{H}, \hat{x}_{ij} > 0\}$, where the capacity of edge (i, j) is $\hat{x}_{ij}$. Using this graph, we compute the maximum flow $\hat{f}_{ij}$ between each pair of distinct vertices $i, j \in \hat{H}$ and determine the potential violation of 2-vertex-connectivity between i and j.

$$\Delta_{ij} = 2(\hat{h}_i - \hat{h}_j - 1) - \hat{f}_{ij}$$

The largest such Δ_{ij} gives us the vertices used in the added constraint (11), while the corresponding saturated edges define the used cut.

4 Experimental results

4.1 Implementation details

The entire solution procedure, including all preprocessing stages, is implemented in Julia 1.12.4. The ILP is solved with Gurobi 12.0.3 through the interface offered by the JuMP library [5]. We keep the default optimality gap of 10^{-4} and run our experiments on an Intel(R) Xeon(R) Silver 4210R processor clocked at 2.40 GHz, with 14 GB of memory. Each run is allowed to use up to 7 parallel threads over a time limit of one hour. This thread limit matters not only for the ILP solver but also for the GRASP heuristic as all operators used in the local search phase take advantage of parallelization. To limit the computational effort devoted to the exploration of these neighborhoods, we impose a time limit equal to 15% of the overall one hour time budget.

At fractional nodes, two procedures can be used: running Algorithm 1 and separating inequalities (11). To avoid a saturation of the model by those constraints, we limit the number of such inequalities to $50|V|$ allowing this limit to scale with the instance size. Algorithm 1 is applied to one out of every two fractional nodes

To draw a fair comparison of the results obtained by this approach and the one of Labbé [4] and Recoba [7], we use the same set of instances taken from the TSPLIB library of Reinelt [8]. To account for multiple cost ratios between edges and arcs, we use the four values $\alpha \in \{3, 5, 7, 9\}$, and define the costs as $c_{ij} = \lceil \alpha \cdot \ell_{ij} \rceil$ for all $(i, j) \in E$, and $d_{ij} = \lceil (10 - \alpha) \cdot \ell_{ij} \rceil$ for all $(i, j) \in A$, where ℓ_{ij} denotes the Euclidean distance between vertices i and j in V.

4.2 Numerical results

For each instance in Table 1, we compare our ILP to the previously best-known solutions for the problem. This solution is either provided by Recoba *et al.*'s heuristic [7] or is given by the optimal Ring-Star structure over the instance computed by Labbé [4]. The column *Previous best solution* displays the minimum cost provided by one of the aforementioned works. The *Improvement* column quantifies the improvement of our solution relative to the previous best, calculated as:

Instance	α	Objective	Best bound	Gap (%)	Node count	Termination status	Solving Time (s.)	Previous best objective value	Improvement (%)
eil51	3	1,278	1,278	0	1	Optimal	7.30	1,278	0
eil51	5	1,995	1,995	0	1	Optimal	8.12	1,995	0
eil51	7	2,100	2,100	0	7,366	Optimal	70.73	2,113	0.62
eil51	9	1,211	1,211	0	1,550	Optimal	11.26	1,224	1.06
berlin52	3	22,626	22,626	0	1	Optimal	11.88	22,626	0
berlin52	5	36,115	36,115	0	352	Optimal	60.24	36,115	0
berlin52	7	37,012	37,010	0	6,841	Optimal	123.96	37,029	0.05
berlin52	9	19,885	19,885	0	10,003	Optimal	20.17	19,887	0.01
st70	3	2,025	2,025	0	1	Optimal	40.80	2,025	0
st70	5	3,110	3,110	0	978	Optimal	271.75	3,110	0
st70	7	3,384	3,351	0.98	1,314,486	Time Limit	TL	3,402	0.53
st70	9	2,398	2,184	8.92	1,142,705	Time Limit	TL	2,513	4.58
pr76	3	319,476	319,476	0	1	Optimal	35.73	324,477	1.54
pr76	5	498,560	498,560	0	10,772	Optimal	1,646.04	500,395	0.37
pr76	7	555,845	531,843	4.32	628,979	Time Limit	TL	555,858	0.002
pr76	9	353,293	323,776	8.35	882,171	Time Limit	TL	378,396	6.63
rat99	3	3,633	3,633	0	1	Optimal	26.45	3,633	0
rat99	5	5,850	5,850	0	2,336	Optimal	1,613.82	5,885	0.59
rat99	7	6,270	5,990	4.47	757,748	Time Limit	TL	6,301	0.49
rat99	9	4,647	3,548	23.65	393,593	Time Limit	TL	4,655	0.17
kroA100	3	63,783	63,783	0	1	Optimal	277.89	63,846	0.1
kroA100	5	101,125	98,490	2.61	2,083	Time Limit	TL	100,785	−0.34
kroA100	7	115,388	105,396	8.66	204,632	Time Limit	TL	115,388	0
kroA100	9	94,996	67,308	29.15	288,065	Time Limit	TL	94,265	−0.77
kroB100	3	66,177	66,177	0	138	Optimal	397.50	66,423	0.37
kroB100	5	104,805	102,765	1.95	2,151	Time Limit	TL	104,550	−0.24
kroB100	7	118,111	107,145	9.28	473,659	Time Limit	TL	118,111	0
kroB100	9	94,948	63,463	33.16	373,464	Time Limit	TL	93,938	−1.08
eil101	3	1,887	1,887	0	1	Optimal	13.61	1,887	0
eil101	5	2,905	2,905	0	1	Optimal	29.21	2,905	0
eil101	7	2,908	2,855	1.82	808,421	Time Limit	TL	2,926	0.62
eil101	9	1,960	1,601	18.32	431,608	Time Limit	TL	1,955	−0.26
bier127	3	354,846	354,846	0	1	Optimal	680.28	354,846	0
bier127	5	542,745	531,000	2.16	2,210	Time Limit	TL	539,955	−0.52
bier127	7	581,095	484,742	16.58	208,697	Time Limit	TL	567,110	−2.47
bier127	9	344,204	267,942	22.16	283,206	Time Limit	TL	343,893	−0.09
ch130	3	18,330	18,330	0	76	Optimal	999.24	18,330	0
ch130	5	29,935	27,605	7.78	1,776	Time Limit	TL	28,679	−4.38
ch130	7	33,371	28,017	16.04	25,824	Time Limit	TL	32,499	−2.68
ch130	9	23,642	16,504	30.19	311,219	Time Limit	TL	23,639	−0.01
kroA150	3	79,572	79,380	0.25	169	Time Limit	TL	79,572	0
kroA150	5	128,985	123,675	4.12	14	Time Limit	TL	125,435	−2.83
kroA150	7	142,520	121,268	14.91	2,810	Time Limit	TL	140,961	−1.11
kroA150	9	114,480	75,040	34.45	66,972	Time Limit	TL	111,882	−2.32
kroB150	3	78,280	77,798	0.62	355	Time Limit	TL	78,390	0.14
kroB150	5	127,510	117,440	7.90	387	Time Limit	TL	122,857	−3.79
kroB150	7	138,076	117,356	15.01	17,905	Time Limit	TL	135,382	−1.99
kroB150	9	108,581	71,190	34.44	104,146	Time Limit	TL	107,925	−0.61

Table 1: Computational results over a subset of the instance of the TSPLIB library

$$100 \times \frac{obj_{prev} - obj_{ilp}}{obj_{prev}},$$

where obj_{prev} is the value in the *Previous best objective value* column, and obj_{ilp} is the final objective value obtained from our proposed branch-and-cut algorithm, as reported in the *Objective* column. A positive value indicates that our model proposes a more competitive solution in terms of objective value, while a negative value signifies no improvement for the current instance.

The *Best bound* and *Gap* columns report the rounded-up value of the best lower bound obtained and the final optimality gap returned by the solver, respectively. The *Node count* column indicates the number of branch-and-cut nodes explored within the time specified in the *Solving time* column, where *TL* denotes

that the time limit of one hour was reached. Finally, the reason for termination is provided in the *Termination status* column.

With respect to the objective value, our approach improves upon the best-known solutions for 17 of the 48 instances. For 14 instances, it matches the results reported in the literature, while it is outperformed in the remaining 17 cases. Among the 14 ties, optimality is proven for 11 instances.

Most improvements occur for instances with 100 vertices or fewer. This trend is closely related to our ability to solve these instances to optimality, as finding optimal solutions for the 2-VC-S is computationally more demanding than for the RSP due to the looser degree constraints imposed on the hubs. Since improvements are primarily observed for instances solved to optimality or near-optimality, this highlights the need for a stronger mathematical formulation to extend these gains to larger instances.

Using GRASP to generate initial solutions allows us to solve two additional instances to optimality compared to the setting without such warm starts. Moreover, for instances that remain unsolved to optimality, a reduction in the optimality gap of up to 7% is observed.

Another avenue for improvement lies in the separation procedure for inequalities (11). Specifically, identifying the maximum violation of these constraints could be optimized using a Gomory-Hu tree [2]. This would reduce the computational complexity of the separation procedure described above from $O(|\hat{H}|^2)$ to $O(|\hat{H}|)$, offering substantial speed-ups for instances in which only a single node is explored. Such instances correspond to low values of α, for which optimal solutions contain only hubs or a majority of hubs. In these solutions, the constraints (11) separated at fractional nodes play an important role in improving the lower bound, because if either of the vertices i or j defining the constraint is not a hub, the constraint is deactivated. Having mostly hubs thus results in a larger number of active constraints.

5 Conclusion

In this work, the 2-Vertex-Connected Star problem is investigated. Since a heuristic algorithm had already been developed to address this problem, we propose an exact approach based on a branch-and-cut framework to complement the previous study. The mathematical formulation is strengthened through the inclusion of flow-based connectivity constraints. Moreover, a primal heuristic based on open-ear decomposition is embedded within the branch-and-cut procedure in order to provide feasible solutions to the solver.

The computational results indicate that adopting an exact approach is beneficial for obtaining solutions of higher quality compared to those reported in the existing literature. In addition, the derived lower bounds provide a formal proof that some solutions proposed in previous related works are indeed optimal for the problem.

As the model exhibits limited scalability with respect to the size and complexity of the instances, future research could focus on improving the quality of the linear programming relaxation, as well as enhancing the efficiency of certain separation techniques. Such efforts could be motivated by the goal of extending the observed improvements in solution quality to larger instances.

References

[1] William R. Pulleyblank Clyde L. Monma, Beth Spellman Munson. 1990. Minimum-weight two-connected spanning networks. *Mathematical Programming* 46 (1990), 153–171. https://doi.org/10.1007/BF01585735

[2] R. E. Gomory and T. C. Hu. 1961. Multi-Terminal Network Flows. *J. Soc. Indust. Appl. Math.* 9, 4 (1961), 551–570. http://www.jstor.org/stable/2098881

[3] Hervé Kerivin and A. Ridha Mahjoub. 2005. Design of Survivable Networks: A survey. *Networks* 46, 1 (2005), 1–21. doi:10.1002/net.20072 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20072

[4] Martine Labbé, Gilbert Laporte, Inmaculada Rodríguez Martín, and Juan José Salazar González. 2004. The Ring Star Problem: Polyhedral analysis and exact algorithm. *Networks* 43, 3 (2004), 177–189. https://onlinelibrary.wiley.com/doi/abs/10.1002/net.10114

[5] Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. 2023. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation* (2023). doi:10.1007/s12532-023-00239-3

[6] A. Lucena and J. E. Beasley. 1998. A branch and cut algorithm for the Steiner problem in graphs. *Networks* 31, 1 (1998), 39–59. https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0037%28199801%2931%3A1%3C39%3A%3AAID-NET5%3E3.0.CO%3B2-L

[7] Rodrigo Recoba, Franco Robledo, Pablo Romero, and Omar Viera. 2018. Two-node-connected star problem. *Int. Trans. Oper. Res.* 25 (2018), 523–543. https://api.semanticscholar.org/CorpusID:33352766

[8] Gerhard Reinelt. 1991. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing* 3, 4 (1991), 376–384. doi:10.1287/ijoc.3.4.376 arXiv:https://doi.org/10.1287/ijoc.3.4.376

[9] Jens M. Schmidt. 2013. A simple test on 2-vertex- and 2-edge-connectivity. *Inform. Process. Lett.* 113, 7 (2013), 241–244. doi:10.1016/j.ipl.2013.01.016

[10] Hassler Whitney. 1932. Non-Separable and Planar Graphs. *Trans. Amer. Math. Soc.* 34, 2 (1932), 339–362. http://www.jstor.org/stable/1989545

[11] Jiefeng Xu, Steve Y. Chiu, and Fred Glover. 1999. Optimizing a Ring-Based Private Line Telecommunication Network Using Tabu Search. *Management Science* 45, 3 (1999), 330–345. http://www.jstor.org/stable/2634881

[12] Abdulhakim Zentani, Nadiatulhuda Zulkifli, and Arnidza Ramli. [n. d.]. Network resiliency and fiber usage of Tree, Star, ring and wheel based wavelength division multiplexed passive optical network Topologies: A comparative review. *Optical Fiber Technology* 73 ([n. d.]), 103038. doi:10.1016/j.yofte.2022.103038