# Integer Programming Formulations and Cutting Plane Algorithms for the Maximum Selective Tree Problem

**Ömer Burak Onar**
Department of Industrial Engineering, Bogazici University, Turkey

**Tınaz Ekim**
Department of Industrial Engineering, Bogazici University, Turkey

**Z. Caner Taşkın**
Department of Industrial Engineering, Bogazici University, Turkey

## Abstract

This paper considers the Maximum Selective Tree Problem (MSelTP) as a generalization of the Maximum Induced Tree problem. Given an undirected graph with a partition of its vertex set into clusters, MSelTP aims to choose the maximum number of vertices such that at most one vertex per cluster is selected and the graph induced by the selected vertices is a tree. To the best of our knowledge, MSelTP has not been studied before although several related optimization problems have been investigated in the literature. We propose two mixed integer programming formulations for MSelTP; one based on connectivity constraints, the other based on cycle elimination constraints. In addition, we develop two exact cutting plane procedures to solve the problem to optimality. On graphs with up to 25 clusters, up to 250 vertices, and varying densities, we conduct computational experiments to compare the results of two solution procedures with solving a compact integer programming formulation of MSelTP. Our experiments indicate that the algorithm CPAXnY outperforms the other procedures overall except for graphs with low density and large cluster size, and that the algorithm CPAX yields better results in terms of the average time of instances optimally solved and the overall average time.

**2012 ACM Subject Classification** Theory of computation → Integer programming; Mathematics of computing → Graph theory; Mathematics of computing → Network optimization

**Keywords and phrases** maximum induced tree, selective tree, cutting plane, separation algorithm, mixed integer programming

**Digital Object Identifier** 10.4230/LIPIcs.SEA.2023.13

**Supplementary Material** *Software (source code)*: `https://github.com/omarburk/MSelTP-Paper`
archived at `swh:1:dir:a1cfaed88df61df35c2a6bf140f4a3ba21af145b`

## 1 Introduction

Given an undirected graph with a partition of its vertex set into clusters, we consider the Maximum Selective Tree Problem (MSelTP) which aims to select at most one vertex per cluster such that the graph induced by the selected vertices is a tree and among all possible vertex selections, the number of vertices of the induced tree is maximized. For problems where alternative decisions are represented by vertices belonging to the same clusters, and selections of vertices (from each cluster) force to consider all edges whose both end vertices are selected, it can be important to obtain a minimally connected graph, which is an induced tree. A water-pipe network, gas-pipe network or a type of circuit having a clustered structure and where the flow uses all pipes or wires among allowed/selected nodes are potential application examples of MSelTP. To the best of our knowledge, MSelTP has not been studied in the literature although various related problems have been considered.

Given a graph, finding a largest vertex subset that induced a tree is called The Maximum Induced Tree problem. This problem is a special case of MSelTP where each cluster in a vertex partition consists of a singleton. Since the decision version of the Maximum Induced Tree problem is NP-complete [11], it follows that MSelTP is also NP-hard. The Maximum Induced Tree problem and several variations of it have been studied from various aspects in the literature [10, 22, 11, 2, 8, 36, 20, 27].

Classical combinatorial optimization problems can often be generalized in various ways. The selective version of each combinatorial optimization problem brings some flexibility into their applications by adding a set of alternatives but also adds in their computational complexity. A selective version consists of taking a graph whose vertices are partitioned into clusters and selecting one vertex per cluster so that the graph induced by the selected vertices admits, among all possible selections, the best solution for the original optimization problem. We note that, in the literature of selective problems, the term cluster is used commonly to mean a set of vertices in a vertex partition. Accordingly, clusters are not complete or dense subgraphs unlike in many other contexts such as community detection. Problems of selective nature have been widely studied in the literature. For instance, the selective graph coloring problem (SelCol) which has been extensively studied [26, 6, 33, 16, 7, 17, 38, 39, 5], takes as input a clustered graph and aims to select exactly one vertex per cluster so that, among all possible such selections, the chromatic number of the graph induced by the selected vertices is minimized. SelCol models the wavelength and routing assignment problem and its selective nature allows us to select a route for each connection from a given set of alternative routes [26].

Generalized network design problems, in general, are obtained by clustered graph instances expressing the feasibility conditions of the classical network design problem in terms of clusters [13]. A selective version of the travelling salesman problem has also been considered in the literature. It aims to select exactly one vertex per cluster so that the selected vertices form a cycle with minimum cost [29, 25, 23].

The generalized spanning tree problem (GMSTP) introduced in [28], takes as input a undirected weighted graph and a partition of its vertex set into clusters, and finds a minimum-cost tree that spans exactly one vertex from each cluster. GMSTP allows us to model local and global networks together for telecommunication networks, where hubs in local networks have to be connected via transmission links such as optical fibers to create a minimum cost global network [28, 12, 14, 4, 19, 30, 15, 21, 34, 18, 32]. The group Steiner tree problem is a more general version of GMSTP [37, 9]. Some other related problems are the prize-collecting generalized minimum spanning tree problem [31] and the generalized traveling salesman problem [3].

Given a graph whose vertex set is partitioned into clusters, another way to generalize classical combinatorial optimization problems is to select representative vertices for each cluster and also edges whose both end vertices are selected and satisfy constraints in the original optimization problem. In this way, the subgraph formed by the selection is not necessarily an induced graph. This notion has been studied in several papers [28, 13, 32, 37, 21, 34, 12, 14, 19, 18, 30, 35, 4, 15, 9, 29, 25, 23], where the number of these representative vertices is required to be at least, at most or exactly one per cluster for different problem types.

Let us now introduce MSelTP in a formal way. MSelTP is defined for an undirected graph $G = (V, E)$ whose vertices are partitioned into $m$ vertex sets called *clusters*. Let $|V| = n$ and $\mathcal{K} = \{V_1, \cdots, V_m\}$ be a clustering of $V$, that is, $V = V_1 \cup V_2 \cdots \cup V_m$ and $V_l \cap V_k = \emptyset$ for all $V_l, V_k \in \mathcal{K}$ such that $l \neq k$. We can assume without loss of generality that edges are defined only between vertices belonging to different clusters since the intracluster edges are irrelevant when at most one vertex is selected from each cluster.

Given a graph $G = (V, E)$ and $V' \subset V$, a graph whose vertex set is $V'$, and whose edges are all the edges of $G$ that have both ends in $V'$ is called a subgraph of $G$ induced by $V'$, and denoted by $G[V']$ [1]. Moreover, for a $|V|$-dimensional binary vector $x$, let $G[x]$ represent the subgraph induced by the vertex set $\{i \in V \mid x_i = 1\}$, which is the set of selected vertices. Similarly, let $E[x]$ denote the edge set of $G[x]$.

MSelTP is the problem of selecting a maximum number of vertices such that at most one vertex is selected per cluster, and the graph induced by the selected vertices is a tree. Such a tree is called a *maximum selective tree*. Figure 1 illustrates a solution for MSelTP in an undirected graph with 15 vertices partitioned into 5 clusters.

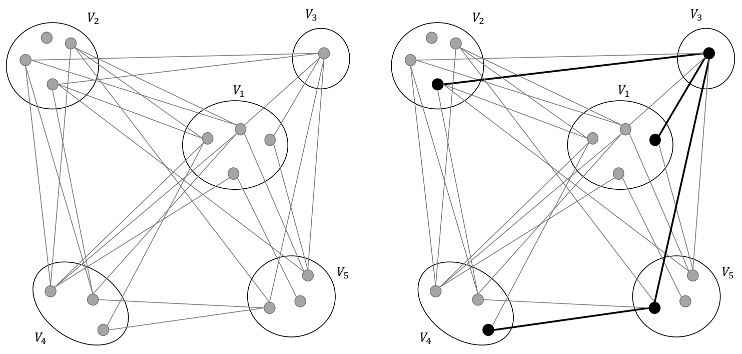


**Figure 1** An optimal solution for an instance of MSelTP.

Given a graph $G = (V, E)$ and a subset $S \subset V$, the set of edges in $E$ that have both endpoints in $S$ is denoted by $E(S)$, and the set of edges in $E$ that have only one endpoint in $S$ is denoted by $\delta(S)$:

$$E(S) = \{(i, j) \in E \mid i \in S,\ j \in S\}$$
$$\delta(S) = \{(i, j) \in E \mid i \in S,\ j \notin S\}$$

Given a graph $G = (V, E)$, the directed graph associated with $G$ is denoted by $D = (V, A)$, whose arc set $A$ is composed of arcs $[i, j]$ and $[j, i]$ for each edge $(i, j) \in E$. Similarly, for a directed graph $D = (V, A)$ and a subset $S \subset V$, the set of arcs in $A$ that have both endpoints in $S$ is denoted by $A(S)$, and the set of arcs in $A$ that have only tail or head in $S$ is denoted as follows:

$$A(S) = \{[i, j] \in A \mid i \in S,\ j \in S\}$$
$$\delta^+(S) = \{[i, j] \in A \mid i \in S,\ j \notin S\}$$
$$\delta^-(S) = \{[i, j] \in A \mid i \notin S,\ j \in S\}$$

We use the notations $\delta^+(i)$ and $\delta^-(i)$ instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ for brevity.

In Section 2, two integer programming formulations of MSelTP are presented: we present an exact formulation, namely flow based formulation (Model 1) in Subsection 2.1, and a formulation with an exponential number of constraints, namely cycle elimination formulation (Model 2) in Subsection 2.2. In Section 3, we develop two cutting plane algorithms, CPAXnY (Algorithm 1) and CPAX (Algorithm 2), based on the cycle elimination formulation. In Section 4, the computational results of the two cutting plane algorithms (CPAXnY and CPAX) and the flow based formulation (FLOW) are compared.

## 2 Formulations for MSelTP

### 2.1 Flow Based Formulation

We propose a formulation for MSelTP based on connectivity constraints. In the formulation, we use flow mechanism introduced by Myung et al. [28] in the directed multicommodity flow model. Model 1 aims to find a connected induced subgraph with maximum number of vertices, which is one more than the number of its edges. Each cluster $V_k \in \mathcal{K}$ corresponds to a commodity. For each commodity $k$, the flow of commodity $k$ should start from the source cluster, which is uniquely designated (for all commodities) to the destination cluster $V_k$. Only one source is designated among all clusters from which a vertex is selected. In order to indicate the direction of flow, we use arcs $[i, j]$ and $[j, i]$ in $A$ instead of each edge $(i, j)$ in $E$, and we create a directed graph $D = (V, A)$ as a directed version of graph $G = (V, E)$. Thus, there may be flow in each direction from $i$ to $j$ and from $j$ to $i$ for each edge $(i, j) \in E$. For each commodity $k$, we introduce non-integer variables $f^k_{ij}$ to represent the flow of commodity $k$ on arc $[i, j]$ and $F^k_i$ to represent the net flow of commodity $k$ through vertex $i$. The following binary variables are introduced:
$x_i = 1$ if the vertex $i$ is selected in the solution, 0 otherwise;
$y_{ij} = 1$ if the edge $(i, j) \in E$ is induced by the selected vertices, 0 otherwise;
$w_{ij} = 1$ if there is flow from the vertex $i \in V$ to the vertex $j \in V$, 0 otherwise;
$s_k = 1$ if the cluster $V_k \in \mathcal{K}$ is designated as the source cluster for all the commodities, 0 otherwise;
$t_k = 1$ if no vertex is selected from the cluster $V_k \in \mathcal{K}$, 0 otherwise.

From a flow related point of view, the binary variable $x_i$ indicates whether the vertex $i$ is included in the network so that all flows are sent through, and $y_{ij}$ indicates if the edge $(i, j)$ is able to carry any flow. The variables $f^k_{ij}$, $F^k_i$, $w_{ij}$ and $s_k$ are auxiliary flow variables. Our flow based formulation is given in Model 1.

Summing all $t$-variables, the objective function minimizes the number of clusters having no selected vertex (1). Constraint (2) ensures that the subgraph defined by any solution has at most one vertex from each cluster and that $t_k$ takes value 1 if no vertex is selected from cluster $V_k$ and 0 otherwise. The number of edges induced by the selected vertices, namely $|E[x]|$, should be exactly one less than the number of selected vertices; this is ensured by constraint (3) since the number of selected vertices equals the number of clusters containing a selected vertex. Two selected vertices force the edge between them to be in the (induced) subgraph, with constraint (4). For an edge to be in the induced subgraph, its both end vertices have to be selected, as forced by constraints (5) and (6). Thus, constraints (4), (5) and (6) ensure that the subgraph defined by any solution is the induced subgraph by the selected vertices. It also allows us to relax $y$-variables as continuous (19) since these constraints force $y$-variables to be integral.

Flow constraints in general include a root vertex/cluster as a source to send all commodities. In MSelTP, the source cluster has to be designated among those clusters including a selected vertex. If the cluster $V_k$ is designated as the source cluster, only then, the variable $s_k$ should be 1. Constraint (7) forces that only one of the clusters is selected as the source cluster and constraint (8) ensures that only the clusters including a selected vertex can be a source cluster.

Constraints (9) are the net flow equations. Let $V_s$ be the source cluster. The LHS of constraint (9) is the net flow of the commodity $k$ over the vertex $i$, which should be 1 if $i$ is the selected vertex in the source cluster $V_s$ unless $k = s$ or $V_k$ does not include any selected vertices; -1 if $i$ is the selected vertex in $V_k$ unless $k = s$; and 0 otherwise. These cases are ensured by constraints (10)-(15). Thus, the net flow constraints (9)-(15) suggest that the subgraph defined by any solution has to be connected.

**Model 1:**

$$\min \sum_{V_k \in \mathcal{K}} t_k \tag{1}$$

s.t.

$$\sum_{i \in V_k} x_i + t_k = 1 \qquad \forall V_k \in \mathcal{K} \tag{2}$$

$$\sum_{(i,j) \in E} y_{ij} = |\mathcal{K}| - 1 - \sum_{V_k \in \mathcal{K}} t_k \tag{3}$$

$$x_i + x_j - 1 \leq y_{ij} \qquad \forall (i,j) \in E \tag{4}$$

$$y_{ij} \leq x_i \qquad \forall (i,j) \in E \tag{5}$$

$$y_{ij} \leq x_j \qquad \forall (i,j) \in E \tag{6}$$

$$\sum_{V_k \in \mathcal{K}} s_k = 1 \tag{7}$$

$$s_k \leq 1 - t_k \qquad \forall V_k \in \mathcal{K} \tag{8}$$

$$\sum_{j:\,[i,j] \in A} f_{ij}^k - \sum_{j:\,[j,i] \in A} f_{ji}^k = F_i^k \qquad \forall i \in V, \quad V_k \in \mathcal{K} \tag{9}$$

$$F_i^k \leq s_k - x_i \qquad \forall i \in V_k, \quad V_k \in \mathcal{K} \tag{10}$$

$$F_i^k \geq s_k - 1 \qquad \forall i \in V_k, \quad V_k \in \mathcal{K} \tag{11}$$

$$F_i^k \leq s_l \qquad \forall i \in V_l, \quad V_l \in \mathcal{K},\ l \neq k \tag{12}$$

$$F_i^k \leq 1 - t_k \qquad \forall i \in V_l, \quad V_l \in \mathcal{K},\ l \neq k \tag{13}$$

$$F_i^k \geq x_i + s_l - t_k - 1 \qquad \forall i \in V_l, \quad V_l \in \mathcal{K},\ l \neq k \tag{14}$$

$$F_i^k \geq 0 \qquad \forall i \in V_l, \quad V_l \in \mathcal{K},\ l \neq k \tag{15}$$

$$f_{ij}^k \leq w_{ij} \qquad \forall [i,j] \in A, \quad V_k \in \mathcal{K} \tag{16}$$

$$w_{ij} + w_{ji} = y_{ij} \qquad \forall (i,j) \in E \tag{17}$$

$$f_{ij}^k \geq 0 \qquad \forall [i,j] \in A, \quad V_k \in \mathcal{K} \tag{18}$$

$$0 \leq y_{ij} \leq 1 \qquad \forall (i,j) \in E \tag{19}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{20}$$

$$w_{ij} \in \{0,1\} \qquad \forall [i,j] \in A \tag{21}$$

$$s_k \in \{0,1\} \qquad \forall V_k \in \mathcal{K} \tag{22}$$

$$t_k \in \{0,1\} \qquad \forall V_k \in \mathcal{K} \tag{23}$$

Considering the edge $(i,j)$ in any selective tree and the flows $f^k_{ij}$ for all $V_k \in \mathcal{K}$ on it, if we hypothetically eliminate this edge to split the selective tree into two trees, for each cluster $V_k$ included in the same tree with the source cluster, the associated commodity $k$ does not flow through $(i,j)$. However, for each cluster $V_k$ included in the other tree, each associated commodity $k$ flows through $(i,j)$ from the tree including the source cluster to the other tree, which indicates that those flows are in the same direction.

When the edge $(i,j)$ is able to carry flow, $w_{ij}$ is hereby used as an indicator of the flow direction for each commodity. Thus, constraints (16) and (17) force that the flow of each commodity on arc $[i,j]$ can be sent only if edge $(i,j)$ is contained in $E[x]$, and if multiple commodities flow on edge $(i,j)$, they all flow in the same direction.

Constraints (5), (6), (9), (16), (17) and (18) ensure that if the vertex $i$ is not selected ($x_i = 0$), related $y$, $w$, $f$ and $F$-variables are forced to be 0 as well. Thus, constraints (12)-(15) allow us to relax $F^k_i$ as $-1 \leq F^k_i \leq 1,\ \forall i \in V,\ V_k \in \mathcal{K}$ since these constraints force $F$-variables to be integral.

Constraint (3) and the connectivity constraints (9, 16, 17) together ensure that the selection induces also a tree. Therefore, each feasible solution corresponds to a selection of an induced tree with at most one vertex per cluster. If the objective function gives the result of $|\mathcal{K}| - 1$, then, the selection is also a generalized spanning tree.

Model 1 has $O((|V| + |E|) \times |\mathcal{K}|)$ constraints, and $O(|V| + |E| + |\mathcal{K}|)$ binary variables out of $O((|V| + |E|) \times |\mathcal{K}|)$ variables. Since it has a polynomial number of variables and constraints, Model 1 is a compact formulation for MSelTP.

## 2.2 Cycle Elimination Formulation

We propose another formulation with constraints eliminating cycles for each vertex subset. This formulation has an exponential number of constraints and $O(|V| + |\mathcal{K}|)$ binary variables. Since an acyclic graph with $n$ vertices and $n - 1$ edges is a tree, Model 2 aims to find an acyclic induced graph with maximum number of vertices, which is one more than the number of its edges. The same set of variables $x_i$, $y_{ij}$ and $t_k$ as in Model 1 are used in Model 2.

**Model 2:**

$$\min \sum_{V_k \in \mathcal{K}} t_k \tag{24}$$

s.t.

$$\sum_{i \in V_k} x_i + t_k = 1 \qquad \forall V_k \in \mathcal{K} \tag{25}$$

$$\sum_{(i,j) \in E} y_{ij} = |\mathcal{K}| - 1 - \sum_{V_k \in \mathcal{K}} t_k \tag{26}$$

$$x_i + x_j - 1 \leq y_{ij} \qquad \forall (i,j) \in E \tag{27}$$

$$y_{ij} \leq x_i \qquad \forall (i,j) \in E \tag{28}$$

$$y_{ij} \leq x_j \qquad \forall (i,j) \in E \tag{29}$$

$$\sum_{(i,j) \in E(S)} y_{ij} \leq |S| - 1 \qquad \forall S \subset V, \quad 3 \leq |S| \tag{30}$$

$$0 \leq y_{ij} \leq 1 \qquad \forall (i,j) \in E \tag{31}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{32}$$

$$t_k \in \{0,1\} \qquad \forall V_k \in \mathcal{K} \tag{33}$$

The flow formulation in Model 1 focuses on the connectivity of the induced subgraph, whereas Model 2 focuses on the fact that the induced graph has to be acyclic. Thus, only the objective function (24), constraint (25) ensuring at most one vertex per cluster, constraint (26) of order-size relation, and constraints (27), (28), (29) forcing the subgraph to be induced by the selected vertices are revisited in this model.

Constraint (30) suggests that the subgraph defined by any solution has to be acylic. For each vertex subset $S \subset V$ with at least 3 vertices, it eliminates all solutions containing a cycle of size $|S|$. Constraints (26) and (30) together ensure that the selection induces a tree. Therefore, each feasible solution yields a selection of an induced tree with at most one vertex per cluster. If the optimal objective function value is $|\mathcal{K}| - 1$, then the selection is also a generalized spanning tree.

## 3 Cutting Plane Methods

Model 2 has an exponential number of constraints, implying that only relatively small instances can be solved with this formulation. Therefore, we develop cutting plane algorithms that add cycle elimination constraints as needed.

Constraint (30) in Model 2 is of exponential cardinality and it ensures that the subgraph defined by any solution is acylic. Since there is an exponential number of constraints (30), we remove them from the formulation and generate them as needed in a cutting plane fashion. In particular, given a selection of vertices, the feasibility of constraint (30) is checked by running a Depth-First Search algorithm to detect cycles. If there is no cycle, then the selection is feasible. Otherwise, let $C$ denote a detected cycle having vertex set $V(C)$ and edge set $E(C)$. We add the following cut:

$$\sum_{(i,j)\in E(C)} y_{ij} \leq |V(C)| - 1 \tag{34}$$

The cutting plane algorithm CPAXnY summarizes the procedure above as:

**Algorithm 1** CPAXnY.

```
Require: A graph $G = (V, E)$ with clustering $\mathcal{K}$
Ensure: An optimal selection $x^*$ of vertices inducing a maximum selective tree
 1: loop
 2:   Solve Model 2 with constraint (30) relaxed. Let $\hat{x}$ be an optimal selection of vertices,
      and $G[\hat{x}]$ be the subgraph induced by $\hat{x}$.
 3:   Find cycles in the induced graph, if any, by depth-first search.
 4:   if $G[\hat{x}]$ does not contain any cycles then
 5:     The selection $\hat{x}$ is optimal.
 6:     $x^* \leftarrow \hat{x}$
 7:     stop
 8:   else
 9:     For some cycle $C$, generate cut (34) and add it to Model 2.
10:   end if
11: end loop
```

Model 2 has $O(|V| + |E| + |\mathcal{K}|)$ variables, which is significantly fewer than in Model 1. Furthermore, we observe that Model 2 can be reformulated in terms of only $x$ and $t$-variables, hence eliminating $y$-variables. To do that, all constraints involving $y$-variables have to be re-expressed. Hence, the related formulation is:

**Model 3:**

$$\min \sum_{V_k \in \mathcal{K}} t_k$$

s.t.

$$\sum_{i \in V_k} x_i + t_k = 1 \qquad \forall V_k \in \mathcal{K} \tag{35}$$

$$|E[x]| = |\mathcal{K}| - 1 - \sum_{V_k \in \mathcal{K}} t_k \tag{36}$$

$$G[x] \text{ admits no cycle} \tag{37}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{38}$$

$$t_k \in \{0,1\} \qquad \forall V_k \in \mathcal{K} \tag{39}$$

Model 3 has $O(|V| + |\mathcal{K}|)$ variables, which is fewer than in Model 1 and Model 2. Since the sum of $y$-variables in constraint (26) indicates the number of edges induced by the selected vertices, the size of $E[x]$, which denotes the edge set of $G[x]$, is used in constraint (36). Since $E[x]$ in (36) depends on the selection and since there is an exponential number of (37), we relax constraints (36) and (37). At any point, if constraint (36) is not satisfied, we need to add a cut to eliminate the current selection from the solution set. Laporte and Louveaux define a binary optimality cut to exclude the current solution [24]. Given binary solution $v^n$ with $S = \{i \mid v_i^n = 1\}$ and $S' = \{i \mid v_i^n = 0\}$, the binary optimality cut is defined as:

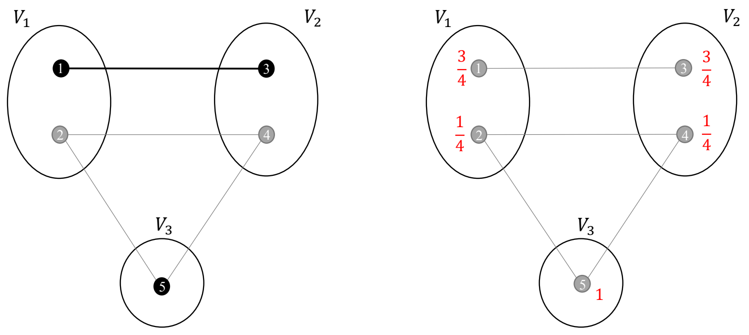
$$\sum_{i \in S} (1 - v_i) + \sum_{i \in S'} v_i \geq 1$$

In our case, the binary variables are $x$ and $t$-variables. Let $\hat{V}$ be the set of selected vertices $\hat{x}$, and $\hat{\mathcal{K}}$ be $\{V_k \in \mathcal{K} \mid \hat{t}_k = 1\}$, indicating the set consisting of clusters, in which no vertex is selected. Thus, in Model 3, the corresponding binary optimality cut is as follows:

$$\sum_{i \in \hat{V}} (1 - x_i) + \sum_{i \in V \setminus \hat{V}} x_i + \sum_{V_k \in \hat{\mathcal{K}}} (1 - t_k) + \sum_{V_k \in \mathcal{K} \setminus \hat{\mathcal{K}}} t_k \geq 1 \tag{40}$$

By constraint (35), changing a 0-valued $x$ or $t$ variable to 1 forces to change a 1-valued variable to 0, and vice versa. Thus, Equation (40) can be improved as follows:

$$\sum_{i \in \hat{V}} (1 - x_i) + \sum_{V_k \in \hat{\mathcal{K}}} (1 - t_k) \geq 1 \tag{41}$$



**(a)** A selection that violates (36). **(b)** A fractional point satisfying cut (40), and violating cut (42).

**Figure 2** A graph example to illustrate that cut (42) is stronger than (40).

Since at most one vertex can be selected from each cluster by constraint (35), the sum of the number of clusters having no selected vertex and the number of selected vertices is equal to the total number of clusters, which leads $|\hat{\mathcal{K}}| + |\hat{V}| = |\mathcal{K}|$. By that equation, an equivalent of Equation (41) is as follows:

$$\sum_{i \in \hat{V}} x_i + \sum_{V_k \in \hat{\mathcal{K}}} t_k \leq |\mathcal{K}| - 1 \tag{42}$$

▶ **Proposition 1.** *Cut* (42) *is stronger than cut* (40).

**Proof.** Any pair $(x, t)$ with $x \geq 0,\ t \geq 0$ satisfying (41) and hence (42) also satisfies (40).

To show that cut (42) is stronger than cut (40), we consider a graph $G = (V, E)$ with clustering $\mathcal{K} = \{V_1, V_2, V_3\}$ where $V = \{1, 2, 3, 4, 5\}$, $V_1 = \{1, 2\}$, $V_2 = \{3, 4\}$, $V_3 = \{5\}$, and $E = \{\{1, 3\}, \{2, 4\}, \{2, 5\}, \{4, 5\}\}$. One possible selection for this graph is $\hat{x} = \{1, 0, 1, 0, 1\}$, hence $\hat{t} = \{0, 0, 0\}$ as in Figure 2a. The induced graph $G[\hat{x}]$ is, then, composed of vertices $\{1, 3, 5\}$ and edge $(1, 3)$, and constraint (36) is not satisfied. This is a case where a binary optimality cut is generated. Cut (40) associated with selection $\hat{x}$ is $(1 - x_1) + (1 - x_3) + (1 - x_5) + x_2 + x_4 + t_1 + t_2 + t_3 \geq 1$, whereas cut (42) associated with selection $\hat{x}$ is $x_1 + x_3 + x_5 \leq 2$. The point $(x, t)$ with $x = (3/4, 1/4, 3/4, 1/4, 1)$ and $t = (0, 0, 0)$, as in Figure 2b, satisfies constraint (35) and cut (40); however, cut (42) is violated. Hence, cut (42) is stronger than cut (40). ◀

Thus, when the number of edges does not satisfy the condition (36), cut (42) is generated. The feasibility of constraint (37) is checked by running a Depth-First Search algorithm to detect cycles. If there is no cycle, then the selection $(x, t)$ is optimum. Otherwise, let $C$ denote a detected cycle having vertex set $V(C)$. We add the following cut:

$$\sum_{i \in V(C)} x_i \leq |V(C)| - 1 \tag{43}$$

The cutting plane algorithm CPAX summarizes the procedure above.

**Algorithm 2** CPAX.

```
Require: A graph $G = (V, E)$ with clustering $\mathcal{K}$
Ensure: An optimal selection $x^*$ of vertices inducing a maximum selective tree
 1: loop
 2:   Solve Model 3 with constraints (36) and (37) relaxed. Let $\hat{x}$ be an optimal selection of
      vertices, and $G[\hat{x}]$ be the subgraph induced by $\hat{x}$.
 3:   Find the subgraph $G[\hat{x}]$ induced by $\hat{x}$.
 4:   Find cycles in $G[\hat{x}]$, if any, by depth-first search.
 5:   if $G[\hat{x}]$ does not contain any cycles then
 6:     if $|E[\hat{x}]| = |\mathcal{K}| - 1 - \sum_{V_k \in \mathcal{K}} \hat{t}_k$ then
 7:       The selection is optimal.
 8:       $x^* \leftarrow \hat{x}$
 9:       stop
10:     else
11:       Generate cut (42) for $(\hat{x}, \hat{t})$ and add it to Model 3.
12:     end if
13:   else
14:     For some cycle $C$, generate cut (43) and add it to Model 3.
15:   end if
16: end loop
```

## 4 Experimental Results

We conduct computational experiments to compare Algorithm 1 and 2 with solving the compact integer programming formulation Model 1 of MSelTP, called FLOW thereafter. We implement the algorithms in C++ and conduct experiments on a computer with an Intel Core i5-7200U @ 2.50 GHz processor and 8 GB RAM with a time limit of 1500 seconds for each experiment. We use CPLEX 20.1 as a solver. We use lazy callback mechanism in CPLEX to apply the cutting plane algorithms. Source code of our implementation is available at `https://github.com/omarburk/MSelTP-Paper`.

We randomly generate test instances as graphs with clusters varying from 5 to 25. We call the number of vertices in a cluster the *cluster size*, and we use three different average cluster size values, which are 3, 6 and 10. As an average density value, we use 0.1, 0.3, 0.5, and 0.7. For each number of clusters, average cluster size and edge density, 10 random instances are generated.

The graphs are randomly generated so that a vertex can be in any cluster with the same probability and for each pair of vertices, the probability of forming an edge between them is the same, which is the edge density value. However, we assume that there is no empty cluster and edges are defined only between vertices belonging to different clusters since the intracluster edges are irrelevant when at most one vertex is selected from each cluster. Additionally, we only choose connected graphs as test instances. In CPAXnY and CPAX, in iterations where multiple cycles are found, we limit the number of cuts to be added in an iteration to 75.

We conduct experiments on 600 test instances of MSelTP to compare the three methods. The number of clusters in these graphs are 5, 10, 15, 20, and 25. Each of them has 3 different sizes as “small”, “medium”, and “large”, corresponding to the average number of vertices per cluster, which are 3, 6, and 10. There are 4 different density options for each case as 0.1 and 0.3 being “low” densities and 0.5 and 0.7 being “high” densities. For each tuple of types,

**Table 1** Summary table for all graph instances.

| cluster size | density | FLOW # opt | FLOW avg gap nonopt | FLOW avg time | CPAXnY # opt | CPAXnY avg gap nonopt | CPAXnY avg time | CPAX # opt | CPAX avg gap nonopt | CPAX avg time |
|---|---|---|---|---|---|---|---|---|---|---|
| small | low | 80 | 83.46 | 331.88 | 100 | | 5.20 | 98 | 61.90 | 63.33 |
| | high | 42 | 76.71 | 902.35 | 100 | | 31.96 | 81 | 31.48 | 372.40 |
| | all | **122** | **78.44** | **617.11** | **200** | | **18.58** | **179** | **34.38** | **217.86** |
| medium | low | 57 | 99.82 | 718.64 | 92 | 63.35 | 147.82 | 89 | 80.28 | 190.67 |
| | high | 31 | 97.63 | 1088.51 | 76 | 61.59 | 485.60 | 58 | 74.35 | 709.72 |
| | all | **88** | **98.47** | **903.57** | **168** | **62.03** | **316.71** | **147** | **75.58** | **450.19** |
| large | low | 46 | 100.00 | 903.01 | 90 | 91.64 | 189.26 | 90 | 92.33 | 178.32 |
| | high | 23 | 99.93 | 1187.03 | 57 | 85.63 | 736.31 | 42 | 88.41 | 896.16 |
| | all | **69** | **99.96** | **1045.02** | **147** | **86.77** | **462.78** | **132** | **88.99** | **537.24** |
| overall | | **279** | **94.21** | **855.24** | **515** | **77.45** | **266.02** | **458** | **75.91** | **401.77** |

10 random cases are generated. We first present the general analysis of the results in Table 1. The cases are grouped by their cluster sizes and densities, in which each combination represents 100 instances. The first two columns in this table identify these combinations. Next columns are split into groups for each method (FLOW, CPAXnY, CPAX). In each column group, the first column shows the number of optimally solved instances out of 100. The second column shows the average of the relative optimality gap for nonoptimal cases in percent within the given time limit of 1500 seconds. The relative optimality gap is calculated as $\left(\frac{UB-LB}{UB} \times 100\right)$, where LB is the best known lower bound on the optimal solution value and UB is the upper bound, which is the best integer objective found. The third column shows the average overall time in seconds within the time limit. For each cluster size group, the last row corresponds to aggregate values for all density groups.

The values at the last row of each group and the final row indicate that CPAXnY outperforms the other two methods in terms of each criteria in the table, except the total average relative gap. Moreover, CPAX yields better results than FLOW. CPAXnY and CPAX optimally solves 86% and 76% of the instances, respectively. FLOW solves only 47% of the instances optimally. In terms of the average relative gap and the average time, the ranking between the methods is the same. For only large-size low-density group of instances, CPAX method seems better than CPAXnY in terms of the average time, despite yielding the average relative gap a bit worse, while their number of optimally solved instances are the same. Moreover, CPAX method seems a bit better than CPAXnY in terms of the total average relative gap although CPAXnY method yields better average relative gaps for each group. The reason is that CPAXnY solves all small-size group optimally, therefore it has no nonoptimal case, which affects the total average relative gap.

We observe that the performance of each method deteriorates as the average cluster size increases. Similarly, as the graph gets denser, the performance of each method becomes worse. While CPAXnY method spends a few seconds on average for small-size low-density group and it solves each one of them to optimality, it rises to 736 seconds for large-size high-density group, and it solves only 57% of the instances optimally.

In order to analyze the results better, we compare the methods separately for different cluster sizes and densities. In our next set of experiments, we compare the results of three methods on graphs with "small", "medium" and "large" cluster sizes. The average number of vertices per cluster is 3 in graphs with "small" cluster size, 6 in graphs with "medium" cluster size, whereas it is 10 in the "large" one. The results of the comparison for graphs with small, medium and large cluster sizes are presented in Tables 2, 3 and 4, respectively in Appendix A.

In each table hereafter, the first three columns in the table identify the number of clusters, the number of vertices, and average edge density across ten random instances. Next columns are split into groups for each method (FLOW, CPAXnY, CPAX). In each column group, the first column shows the number of optimally solved instances out of 10. The second column shows the average of the relative optimality gap in percent of the instances that are not solved to optimality within the given time limit of 1500 seconds, while the third column shows that of all the instances. The relative optimality gap is calculated as $\left(\frac{UB-LB}{UB} \times 100\right)$, where LB is the best known lower bound on the optimal solution value and UB is the upper bound, which is the best integer objective found. The fourth and the fifth columns show the average time in seconds of the instances that are not solved to optimality within the time limit and that of all the instances, respectively. The procedures CPAXnY and CPAX have one last column that shows average time spent in their subproblem in seconds of all the instances. As we can see from Tables 2, 3 and 4, in each row, the average time in subproblems is negligible. All the values in each row are the average result of the 10 instances for that case.

As we can see the results in Table 2, the algorithm CPAXnY outperforms the other procedures in terms of the number of optimally solved instances, the average of the relative optimality gap for both nonoptimal cases and overall, and the average time of instances optimally solved and the overall average time. For graphs with lower density and a smaller number of cluster, its results are similar with that of CPAX in terms of the average time of instances optimally solved and the overall average time, yet, as the number of clusters increases, it gave better results. Furthermore, for graphs with higher density, CPAXnY yields significantly better results than others. It still optimally solves all the cases whereas CPAX is not able to solve to optimality except one instance for graphs with 25 clusters. Moreover, for graphs with 20 and 25 clusters, the overall average time of CPAX is longer more than 10 times that of CPAXnY. Both outperform FLOW significantly, which is able to solve very few instances optimally as the number of clusters increases to 15 for graphs with higher density.

As we can see the results in Table 3, the algorithm CPAXnY still outperforms the other procedures in terms of each criteria. However, the performance of each method deteriorates as the cluster size increases. For graphs with lower density and fewer number of clusters, the results of CPAXnY are similar with that of CPAX in terms of the average time of instances optimally solved and the overall average time, yet, as the number of clusters increases, it gave better results. This time, instead of all cases, CPAXnY and CPAX optimally solve 92% and 89% of the instances, respectively. Furthermore, for graphs with higher density, CPAXnY yields much better results than others. However, it optimally solves 76% of the instances this time and 80% of the instances for graphs with 20 cluster, whereas CPAX is not able to solve any instances to optimality for graphs with 20 and 25 clusters. Moreover, CPAXnY has better average of the relative optimality gap for both nonoptimal cases and overall. Both outperform FLOW significantly. Among graphs with higher density, it is able to solve the instances optimally for only those with 5 clusters and half of those with 10 clusters.

As we can see the results in Table 4, for graphs with lower density, the algorithm CPAXnY outperforms the other procedures in terms of the average of the relative optimality gap for both nonoptimal cases and overall; however, the algorithm CPAX yields better results than the other procedures in terms of the average time of instances optimally solved and the overall average time. CPAXnY and CPAX solve the same number of instances to optimality. For graphs with lower density, both CPAXnY and CPAX optimally solve 90% of the instances. Nevertheless, for graphs with higher density, CPAXnY yields much better results than others in terms of each criteria. It optimally solves 57% of the instances this time and 85% of the instances for graphs with 15 cluster, whereas CPAX optimally solves 42% of the instances and it is not able to solve any instances to optimality for graphs with 15, 20 and 25 clusters except 3 instances for those with 15. Their average times are closer than before. Moreover, CPAXnY has better average of the relative optimality gap for both nonoptimal cases and overall. Both outperform FLOW significantly. Among graphs with higher density, it is able to solve the instances optimally for only those with 5 clusters and 15% of those with 10 clusters. As a limitation, for the large-size high-density graph instances, CPAXnY is not able to solve after those with 20 clusters in 1500 seconds, whereas CPAX is not almost able to solve after those with 15 clusters in 1500 seconds and FLOW is not after those with 10 clusters. As the number of clusters or density increases, the performance of all methods deteriorates in general, especially FLOW method worsens faster. Intuitively, one can think that finding a tree gets easier as density increases; however, finding an induced tree gets harder since the possibility of including a cycle in an induced subgraph increases. Comparing Tables 2, 3 and 4, we observe also that for graphs with high density, the performance of all three methods gets worse in general as the cluster size increases. However, for graphs with low density, the deterioration in performance is less than high density graphs.

As a future research direction, MSelTP can be examined for different kinds of graph classes in terms of algorithms, formulations, and complexity. Moreover, a variant of MSelTP can be studied for forests instead of trees; this would correspond to the selective version of the well-known feedback vertex set problem, which has not been studied to the best of our knowledge. Additionally, vertex-weighted and/or edge-weighted versions of MSelTP can be studied.

## References

1 John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph Theory with Applications*. Elsevier Science Publishing Co., Inc., 1976.

2 M. Chudnovsky and Paul D. Seymour. The three-in-a-tree problem. *Combinatorica*, 30:387–417, 2010.

3 Ovidiu Cosma, Petrică C. Pop, and Laura Cosma. An effective hybrid genetic algorithm for solving the generalized traveling salesman problem. In Hugo Sanjurjo González, Iker Pastor López, Pablo García Bringas, Héctor Quintián, and Emilio Corchado, editors, *Hybrid Artificial Intelligent Systems*, pages 113–123, Cham, 2021. Springer International Publishing.

4 Ernando Gomes de Sousa, Rafael Castro de Andrade, and Andréa C. Santos. A multigraph formulation for the generalized minimum spanning tree problem. In *ISCO 2018: Combinatorial Optimization*, volume 10856 of *Lecture Notes in Computer Science*, pages 133–143. Springer, July 2018. doi:10.1007/978-3-319-96151-4_12.

5 Marc Demange, Tınaz Ekim, and Bernard Ries. On the minimum and maximum selective graph coloring problems in some graph classes. *Discrete Applied Mathematics*, 204:77–89, 2016. doi:10.1016/j.dam.2015.10.005.

6 Marc Demange, Tınaz Ekim, Bernard Ries, and Cerasela Tanasescu. On some applications of the selective graph coloring problem. *European Journal of Operational Research*, 240(2):307–314, 2015. doi:10.1016/j.ejor.2014.05.011.

7 Marc Demange, Jérôme Monnot, Petrica Pop, and Bernard Ries. On the complexity of the selective graph coloring problem in some special classes of graphs. *Theoretical Computer Science*, 540-541:89–102, 2014. Combinatorial Optimization: Theory of algorithms and Complexity. doi:10.1016/j.tcs.2013.04.018.

8 Nicolas Derhy and Christophe Picouleau. Finding induced trees. *Discrete Applied Mathematics*, 157:3552–3557, October 2009. doi:10.1016/j.dam.2009.02.009.

9 C.W Duin, A Volgenant, and S Voß. Solving group Steiner problems as Steiner problems. *European Journal of Operational Research*, 154(1):323–329, 2004. doi:10.1016/S0377-2217(02)00707-5.

10 Paul Erdös and Zbigniew Palka. Trees in random graphs. *Discret. Math.*, 46:145–150, 1983.

11 Paul Erdös, Michael Saks, and Vera T Sós. Maximum induced trees in graphs. *Journal of Combinatorial Theory, Series B*, 41(1):61–79, 1986. doi:10.1016/0095-8956(86)90028-6.

12 Corinne Feremans, Martine Labbé, and Gilbert Laporte. A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks*, 39:29–34, January 2002. doi:10.1002/net.10009.

13 Corinne Feremans, Martine Labbé, and Gilbert Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1–13, 2003. doi:10.1016/S0377-2217(02)00404-6.

14 Corinne Feremans, Martine Labbé, Gilbert Laporte, and Etudes Commerciales. The generalized minimum spanning tree problem: Polyhedral analysis and branch-and-cut algorithm. *Networks*, 43, November 2002. doi:10.1002/net.10105.

15 Cristiane S. Ferreira, Luis Satoru Ochi, Víctor Parada, and Eduardo Uchoa. A GRASP-based approach to the generalized minimum spanning tree problem. *Expert Systems with Applications*, 39(3):3526–3536, 2012. doi:10.1016/j.eswa.2011.09.043.

16 Yuri Frota, Nelson Maculan, Thiago F Noronha, and Celso C Ribeiro. A branch-and-cut algorithm for partition coloring. *Networks: An International Journal*, 55(3):194–204, 2010.

17 Fabio Furini, Enrico Malaguti, and Alberto Santini. An exact algorithm for the partition coloring problem. *Computers & Operations Research*, 92:170–181, 2018. doi:10.1016/j.cor.2017.12.019.

18 Bruce Golden, Saahitya Raghavan, and Daliborka Stanojevic. The prize-collecting generalized minimum spanning tree problem. *Journal of Heuristics*, 14:69–93, February 2008. doi:10.1007/s10732-007-9027-1.

19 Mohamed Haouari and Jouhaina Siala. Upper and lower bounding strategies for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 171:632–647, February 2006. doi:10.1016/j.ejor.2004.07.072.

20 Alain Hertz, Odile Marcotte, and David Schindl. On the maximum orders of an induced forest, an induced tree, and a stable set. *Yugoslav Journal of Operations Research*, 24:199–215, January 2014. doi:10.2298/YJOR130402037H.

21 Edmund Ihler, Gabriele Reich, and Peter Widmayer. Class Steiner trees and VLSI-design. *Discrete Applied Mathematics*, 90(1):173–194, 1999. doi:10.1016/S0166-218X(98)00090-0.

22 Michal Karonski and Zbigniew Palka. On the size of a maximal induced tree in a random graph. *Math. Slovaca*, 30:151–155, 1980.

23 Gilbert Laporte, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. Some applications of the generalized travelling salesman problem. *The Journal of the Operational Research Society*, 47(12):1461–1467, 1996.

24 Gilbert Laporte and François V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993. doi:10.1016/0167-6377(93)90002-X.

25 Gilbert Laporte and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR: Information Systems and Operational Research*, 21(1):61–75, 1983. doi:10.1080/03155986.1983.11731885.

26 Guangzhi Li and Rahul Simha. The partition coloring problem and its application to wavelength routing and assignment. In *In Proceedings of the First Workshop on Optical Networks*, 2000.

27 Rafael A. Melo and Celso C. Ribeiro. Maximum weighted induced forests and trees: New formulations and a computational comparative review. *International Transactions in Operational Research*, 2021. doi:10.1111/itor.13066.

28 Young Soo Myung, Chang Ho Lee, and Dong Wan Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995. doi:10.1002/net.3230260407.

29 Charles E. Noon and James C. Bean. A lagrangian based approach for the asymmetric generalized traveling salesman problem. *Operations Research*, 39(4):623–632, 1991.

30 Temel Oncan, Jean-François Cordeau, and Gilbert Laporte. A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 191:306–319, December 2008. doi:10.1016/j.ejor.2007.08.021.

31 Petrica C. Pop. On the prize-collecting generalized minimum spanning tree problem. *Annals of Operations Research*, 150:193–204, March 2007. doi:10.1007/s10479-006-0153-1.

32 Petrica C. Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 283(1):1–15, 2020. doi:10.1016/j.ejor.2019.05.017.

33 Petrica C. Pop, Bin Hu, and Günther Raidl. A memetic algorithm with two distinct solution representations for the partition graph coloring problem. In *Revised Selected Papers of the 14th International Conference on Computer Aided Systems Theory - EUROCAST 2013 - Volume 8111*, volume 8111, pages 219–226. Springer-Verlag, February 2013. doi:10.1007/978-3-642-53856-8_28.

34 Petrica C. Pop, Walter Kern, and Georg J. Still. *The Generalized Minimum Spanning Tree Problem*. University of Twente, Department of Applied Mathematics, 2000.

35 Petrica C. Pop, Oliviu Matei, Cosmin Sabo, and Adrian Petrovan. A two-level solution approach for solving the generalized minimum spanning tree problem. *European Journal of Operational Research*, August 2017. doi:10.1016/j.ejor.2017.08.015.

36 Dieter Rautenbach. Dominating and large induced trees in regular graphs. *Discrete Mathematics*, 307(24):3177–3186, 2007. doi:10.1016/j.disc.2007.03.043.

37 Gabriele Reich and Peter Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. In *Graph-Theoretic Concepts in Computer Science*, volume 411 of *Lecture Notes in Computer Science*, pages 196–210. Springer, Berlin, Heidelberg, 1990. doi:10.1007/3-540-52292-1_14.

38 Oylum Şeker, Tınaz Ekim, and Z. Caner Taşkın. A decomposition approach to solve the selective graph coloring problem in some perfect graph families. *Networks*, 73(2):145–169, 2019. doi:10.1002/net.21850.

39 Oylum Şeker, Tınaz Ekim, and Z. Caner Taşkın. An exact cutting plane algorithm to solve the selective graph coloring problem in perfect graphs. *European Journal of Operational Research*, 291(1):67–83, 2021. doi:10.1016/j.ejor.2020.09.017.

## A Computational Results for graphs with small/medium/large cluster sizes

**Table 2** Computational results for graphs with small cluster size.

| | | | FLOW | | | | | CPAXnY | | | | | | CPAX | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # clust | # vert | avg density | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr |
| 5 | 15 | 0.1 | 10 | | 0.00 | 0.22 | 0.22 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 | 10 | | 0.00 | 0.11 | 0.11 | 0.01 |
| 5 | 15 | 0.3 | 10 | | 0.00 | 0.25 | 0.25 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 |
| 10 | 30 | 0.1 | 10 | | 0.00 | 0.71 | 0.71 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 3.95 | 3.95 | 0.09 |
| 10 | 30 | 0.3 | 10 | | 0.00 | 2.94 | 2.94 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 15 | 45 | 0.1 | 10 | | 0.00 | 1.74 | 1.74 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 | 10 | | 0.00 | 26.60 | 26.60 | 0.40 |
| 15 | 45 | 0.3 | 10 | | 0.00 | 134.85 | 134.85 | 10 | | 0.00 | 0.52 | 0.52 | 0.00 | 10 | | 0.00 | 0.06 | 0.06 | 0.01 |
| 20 | 60 | 0.1 | 10 | | 0.00 | 31.11 | 31.11 | 10 | | 0.00 | 0.26 | 0.26 | 0.01 | 10 | | 0.00 | 36.27 | 36.27 | 0.44 |
| 20 | 60 | 0.3 | 0 | 77.03 | 77.03 | | 1500.00 | 10 | | 0.00 | 3.93 | 3.93 | 0.01 | 10 | | 0.00 | 12.59 | 12.59 | 0.14 |
| 25 | 75 | 0.1 | 10 | | 0.00 | 146.96 | 146.96 | 10 | | 0.00 | 6.52 | 6.52 | 0.04 | 9 | 100.00 | 10.00 | 6.31 | 155.68 | 0.26 |
| 25 | 75 | 0.3 | 0 | 89.90 | 89.90 | | 1500.00 | 10 | | 0.00 | 40.64 | 40.64 | 0.02 | 9 | 23.81 | 2.38 | 275.48 | 397.93 | 0.52 |
| | | | **80** | **83.46** | **16.69** | **39.85** | **331.88** | **100** | **0.00** | **0.00** | **5.20** | **5.20** | **0.01** | **98** | **61.90** | **1.24** | **34.01** | **63.33** | **0.18** |
| 5 | 15 | 0.5 | 10 | | 0.00 | 0.21 | 0.21 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 5 | 15 | 0.7 | 10 | | 0.00 | 0.26 | 0.26 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 10 | 30 | 0.5 | 10 | | 0.00 | 39.64 | 39.64 | 10 | | 0.00 | 0.40 | 0.40 | 0.00 | 10 | | 0.00 | 0.05 | 0.05 | 0.01 |
| 10 | 30 | 0.7 | 10 | | 0.00 | 53.75 | 53.75 | 10 | | 0.00 | 0.86 | 0.86 | 0.00 | 10 | | 0.00 | 0.24 | 0.24 | 0.01 |
| 15 | 45 | 0.5 | 2 | 52.31 | 41.84 | 1148.08 | 1429.62 | 10 | | 0.00 | 2.65 | 2.65 | 0.00 | 10 | | 0.00 | 5.65 | 5.65 | 0.07 |
| 15 | 45 | 0.7 | 0 | 64.88 | 64.88 | | 1500.00 | 10 | | 0.00 | 5.21 | 5.21 | 0.00 | 10 | | 0.00 | 19.54 | 19.54 | 0.12 |
| 20 | 60 | 0.5 | 0 | 80.89 | 80.89 | | 1500.00 | 10 | | 0.00 | 27.18 | 27.18 | 0.00 | 10 | | 0.00 | 322.85 | 322.85 | 0.43 |
| 20 | 60 | 0.7 | 0 | 79.16 | 79.16 | | 1500.00 | 10 | | 0.00 | 24.42 | 24.42 | 0.00 | 10 | | 0.00 | 428.15 | 428.15 | 0.57 |
| 25 | 75 | 0.5 | 0 | 88.40 | 88.40 | | 1500.00 | 10 | | 0.00 | 154.78 | 154.78 | 0.00 | 1 | 32.97 | 29.67 | 974.91 | 1447.49 | 0.74 |
| 25 | 75 | 0.7 | 0 | 89.75 | 89.75 | | 1500.00 | 10 | | 0.00 | 104.01 | 104.01 | 0.00 | 0 | 30.14 | 30.14 | | 1500.00 | 0.00 |
| | | | **42** | **76.71** | **44.49** | **77.02** | **902.35** | **100** | **0.00** | **0.00** | **31.96** | **31.96** | **0.00** | **81** | **31.48** | **5.98** | **107.90** | **372.40** | **0.16** |

**Table 3** Computational results for graphs with medium cluster size.

| | | | FLOW | | | | | CPAXnY | | | | | | CPAX | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # clust | # vert | avg density | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr |
| 5 | 30 | 0.1 | 10 | | 0.00 | 0.20 | 0.20 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.09 | 0.09 | 0.01 |
| 5 | 30 | 0.3 | 10 | | 0.00 | 0.44 | 0.44 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 10 | 60 | 0.1 | 10 | | 0.00 | 2.53 | 2.53 | 10 | | 0.00 | 0.07 | 0.07 | 0.00 | 10 | | 0.00 | 49.45 | 49.45 | 0.56 |
| 10 | 60 | 0.3 | 10 | | 0.00 | 35.46 | 35.46 | 10 | | 0.00 | 0.13 | 0.13 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 15 | 90 | 0.1 | 10 | | 0.00 | 52.15 | 52.15 | 10 | | 0.00 | 0.22 | 0.22 | 0.00 | 9 | 100.00 | 10.00 | 8.66 | 157.80 | 0.20 |
| 15 | 90 | 0.3 | 2 | 99.46 | 79.57 | 875.46 | 1375.09 | 10 | | 0.00 | 1.72 | 1.72 | 0.00 | 10 | | 0.00 | 0.25 | 0.25 | 0.03 |
| 20 | 120 | 0.1 | 5 | 100.00 | 50.00 | 941.10 | 1220.55 | 10 | | 0.00 | 0.67 | 0.67 | 0.01 | 10 | | 0.00 | 6.37 | 6.37 | 0.20 |
| 20 | 120 | 0.3 | 0 | 99.75 | 99.75 | | 1500.00 | 10 | | 0.00 | 99.67 | 99.67 | 0.03 | 10 | | 0.00 | 192.18 | 192.18 | 0.89 |
| 25 | 150 | 0.1 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 2.81 | 2.81 | 0.03 | 10 | | 0.00 | 0.48 | 0.48 | 0.09 |
| 25 | 150 | 0.3 | 0 | 99.89 | 99.89 | | 1500.00 | 2 | 63.35 | 50.68 | 864.13 | 1372.83 | 0.04 | 0 | 78.30 | 78.30 | | 1500.00 | 0 |
| | | | **57** | **99.82** | **42.92** | **129.19** | **718.64** | **92** | **63.35** | **5.07** | **30.24** | **147.82** | **0.01** | **89** | **80.28** | **8.83** | **28.84** | **190.67** | **0.22** |
| 5 | 30 | 0.5 | 10 | | 0.00 | 0.74 | 0.74 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 5 | 30 | 0.7 | 10 | | 0.00 | 1.85 | 1.85 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 10 | 60 | 0.5 | 10 | | 0.00 | 506.90 | 506.90 | 10 | | 0.00 | 0.98 | 0.98 | 0.00 | 10 | | 0.00 | 0.35 | 0.35 | 0.02 |
| 10 | 60 | 0.7 | 1 | 90.26 | 81.23 | 256.02 | 1375.60 | 10 | | 0.00 | 4.19 | 4.19 | 0.00 | 10 | | 0.00 | 7.12 | 7.12 | 0.11 |
| 15 | 90 | 0.5 | 0 | 97.41 | 97.41 | | 1500.00 | 10 | | 0.00 | 34.52 | 34.52 | 0.00 | 10 | | 0.00 | 222.16 | 222.16 | 0.60 |
| 15 | 90 | 0.7 | 0 | 97.73 | 97.73 | | 1500.00 | 10 | | 0.00 | 66.62 | 66.62 | 0.00 | 8 | 45.00 | 9.00 | 709.40 | 867.52 | 1.30 |
| 20 | 120 | 0.5 | 0 | 98.95 | 98.95 | | 1500.00 | 8 | 37.18 | 7.44 | 699.24 | 832.71 | 0.01 | 0 | 64.00 | 64.00 | | 1500.00 | 0.00 |
| 20 | 120 | 0.7 | 0 | 98.33 | 98.33 | | 1500.00 | 8 | 24.62 | 4.92 | 771.07 | 916.86 | 0.01 | 0 | 63.57 | 63.57 | | 1500.00 | 0.00 |
| 25 | 150 | 0.5 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 73.36 | 73.36 | | 1500.00 | 0.00 | 0 | 87.22 | 87.22 | | 1500.00 | 0.00 |
| 25 | 150 | 0.7 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 62.08 | 62.08 | | 1500.00 | 0.00 | 0 | 88.47 | 88.47 | | 1500.00 | 0.00 |
| | | | **31** | **97.63** | **67.37** | **172.61** | **1088.51** | **76** | **61.59** | **14.78** | **168.77** | **485.60** | **0.00** | **58** | **74.35** | **31.23** | **137.45** | **709.72** | **0.31** |

**Table 4** Computational results for graphs with large cluster size.

| | | | FLOW | | | | | CPAXnY | | | | | | CPAX | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # clust | # vert | avg density | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr | # opt | avg % gap in nonopt | avg % gap overall | avg time in opt | avg time overall | avg time in subpr |
| 5 | 50 | 0.1 | 10 | | 0.00 | 0.78 | 0.78 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 | 10 | | 0.00 | 0.39 | 0.39 | 0.03 |
| 5 | 50 | 0.3 | 10 | | 0.00 | 1.81 | 1.81 | 10 | | 0.00 | 0.05 | 0.05 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 10 | 100 | 0.1 | 10 | | 0.00 | 8.10 | 8.10 | 10 | | 0.00 | 0.17 | 0.17 | 0.00 | 10 | | 0.00 | 107.83 | 107.83 | 0.82 |
| 10 | 100 | 0.3 | 10 | | 0.00 | 356.09 | 356.09 | 10 | | 0.00 | 0.51 | 0.51 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 15 | 150 | 0.1 | 6 | 100.00 | 40.00 | 938.88 | 1163.33 | 10 | | 0.00 | 0.86 | 0.86 | 0.00 | 10 | | 0.00 | 4.99 | 4.99 | 0.15 |
| 15 | 150 | 0.3 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 10.90 | 10.90 | 0.01 | 10 | | 0.00 | 0.16 | 0.16 | 0.03 |
| 20 | 200 | 0.1 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 5.59 | 5.59 | 0.01 | 10 | | 0.00 | 1.73 | 1.73 | 0.15 |
| 20 | 200 | 0.3 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 345.68 | 345.68 | 0.06 | 10 | | 0.00 | 164.77 | 164.77 | 1.32 |
| 25 | 250 | 0.1 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 28.77 | 28.77 | 0.03 | 10 | | 0.00 | 3.31 | 3.31 | 0.26 |
| 25 | 250 | 0.3 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 91.64 | 91.64 | | 1500.00 | 0.00 | 0 | 92.33 | 92.33 | | 1500.00 | 0.00 |
| | | | **46** | **100.00** | **54.00** | **202.20** | **903.01** | **90** | **91.64** | **9.16** | **43.62** | **189.26** | **0.01** | **90** | **92.33** | **9.23** | **31.47** | **178.32** | **0.31** |
| 5 | 50 | 0.5 | 10 | | 0.00 | 2.84 | 2.84 | 10 | | 0.00 | 0.06 | 0.06 | 0.00 | 10 | | 0.00 | 0.02 | 0.02 | 0.00 |
| 5 | 50 | 0.7 | 10 | | 0.00 | 6.65 | 6.65 | 10 | | 0.00 | 0.07 | 0.07 | 0.00 | 10 | | 0.00 | 0.03 | 0.03 | 0.00 |
| 10 | 100 | 0.5 | 3 | 100.00 | 70.00 | 1036.14 | 1360.84 | 10 | | 0.00 | 2.53 | 2.53 | 0.00 | 10 | | 0.00 | 0.27 | 0.27 | 0.02 |
| 10 | 100 | 0.7 | 0 | 99.93 | 99.93 | | 1500.00 | 10 | | 0.00 | 26.70 | 26.70 | 0.00 | 9 | 83.33 | 8.33 | 159.36 | 293.43 | 0.58 |
| 15 | 150 | 0.5 | 0 | 100.00 | 100.00 | | 1500.00 | 10 | | 0.00 | 465.66 | 465.66 | 0.01 | 3 | 72.86 | 51.00 | 392.90 | 1167.87 | 1.47 |
| 15 | 150 | 0.7 | 0 | 99.52 | 99.52 | | 1500.00 | 7 | 44.44 | 13.33 | 701.94 | 868.05 | 0.00 | 0 | 73.24 | 73.24 | | 1500.00 | 0.00 |
| 20 | 200 | 0.5 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 87.30 | 87.30 | | 1500.00 | 0.00 | 0 | 90.57 | 90.57 | | 1500.00 | 0.00 |
| 20 | 200 | 0.7 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 81.55 | 81.55 | | 1500.00 | 0.00 | 0 | 92.46 | 92.46 | | 1500.00 | 0.00 |
| 25 | 250 | 0.5 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 94.44 | 94.44 | | 1500.00 | 0.00 | 0 | 99.82 | 99.82 | | 1500.00 | 0.00 |
| 25 | 250 | 0.7 | 0 | 100.00 | 100.00 | | 1500.00 | 0 | 91.60 | 91.60 | | 1500.00 | 0.00 | 0 | 97.35 | 97.35 | | 1500.00 | 0.00 |
| | | | **23** | **99.93** | **76.94** | **139.28** | **1187.03** | **57** | **85.63** | **36.82** | **173.05** | **736.31** | **0.00** | **42** | **88.41** | **51.28** | **62.29** | **896.16** | **0.23** |