Polynomial time algorithm for a Bi-objective Spanning Star Forest Problem on Trees

Thanh Loan Nguyen[1] and Viet Hung Nguyen[1]*

INP Clermont Auvergne, Univ Clermont Auvergne, Mines Saint-Etienne, CNRS, UMR 6158 LIMOS, 1 Rue de la Chebarde, 63178 Aubiere Cedex, France
thanh_loan.nguyen@doctorant.uca.fr, viet_hung.nguyen@uca.fr

Abstract. A star is a graph in which all edges share a common endpoint, called the center, and a star forest is a collection of disjoint stars. For a connected weighted graph, a spanning star forest (SSF) is a star forest that spans all nodes of the graph. The total weight of a SSF is the sum of its edge weights in the weighted case, or the number of its edges in the unweighted case. Recently, the minimum weight spanning star forest problem with a fixed number of centers has attracted attention. In the context of the automobile industry, Agra et al. [1] studied the problem with at most k centers and proved it NP-hard, while Briant and Naddef [5] investigated a variant with exactly k centers. However, both approaches require the value of k to be given in advance, with no criterion for selecting it. To overcome this limitation, we introduce the Bi-objective Spanning Star Forest Problem (BSSFP), which jointly minimizes the number of centers and the total edge weight. Since the version with bounded k centers is NP-hard, the BSSFP is also NP-hard on general graphs. To our knowledge, no efficient method exists for enumerating Pareto-optimal SSFs that consider both objectives, even in restricted graph classes. We focus on the case where the input graph is a tree and prove that all efficient solutions of the BSSFP can be enumerated using a two-phase dynamic programming algorithm. This provides the first polynomial-time algorithm for solving the BSSFP on trees.

Keywords: Bi-objective Combinatorial Optimization · Spanning Star Forest · Automobile industry · Dynamic programming

1 Introduction

Given an undirected graph $G = (V, E)$, a *star* in G is defined as either an isolated node or a subgraph where all edges share a common endpoint called the *center*. A *star forest* is a collection of disjoint stars. A *spanning star forest* (SSF) of G is a star forest that contains all nodes of G. Let $w \in \mathbb{R}_+^{|E|}$ be the weight vector associated with edges in G. The total weight of a SSF is the sum of its edge weights in the weighted case, and it is the number of its edges in the unweighted case. The single objective problem of finding a maximum weight

* Corresponding author.

star forest is a combinatorial optimization problem that has received significant attention in recent years. C. T. Nguyen et al. [10] prove that the problem is APX-hard. Moreover, they developed a linear-time algorithm for solving this problem on trees, leading to a $\frac{1}{2}$-approximation algorithm for general graphs. Recently, V. H. Nguyen [12] extended this approach to cactus graphs by introducing a polynomial-time algorithm for this class.

In contrast, the minimum weight star forest of SSF is trivial since a solution where every node is isolated already forms a spanning star forest with total weight zero and $|V|$ centers. Thus, the minimum version is significant when the number of centers is bounded. This version is motivated by practical applications in the automobile industry. Agra et al. [1] considered the problem of automobile option selection systems, where each center corresponds to a base configuration, and showed that minimizing the weight of configurations with at most k centers is NP-hard on general graphs by reduction from the 3-minimum cover problem. They proposed an integer programming formulation and heuristic algorithms. In a related work, [5] introduced the Optimal Diversity Management Problem (ODMP), which requires choosing exactly k models (centers) and replacing every other model by a compatible one of minimum weight. They proved that ODMP is NP-hard in general cases, as it can be formulated as a p-median problem on a partial-order graph. Their solution strategy is based on Lagrangian relaxation and variable fixing, tackling the solution of large instances. Observe that in these two works, the value of k must be known before solving the problem. However, this raises a natural question: what criterion should be used to select k?

To avoid committing to a single value of k and to explore the full trade-off, we introduce a minimum bi-objective version of this problem, which is called *Bi-objective Spanning Star Forest Problem (BSSFP)*, where both the number of centers and the total edge weight are minimized. In the bi-objective version, no constraint on the value of k needs to be specified, which avoids the need to predefine k and allows exploration of trade-offs. For the general graph, as both the minimum weight SSF problem with a bound on the number of centers and the problem of minimum number of centers (which corresponds to maximizing the total weight in the unweighted case) are NP-hard, it follows that the BSSFP is also NP-hard. In bi-objective optimization, feasible solutions are compared under Pareto dominance. A solution is called *efficient* (i.e., Pareto optimal) if no other solution improves one objective without worsening the other. Efficient solutions are typically classified as either *supported efficient solutions*, which can be obtained by solving scalarized single-objective problems such as weighted sums and lying on the boundary of the convex hull of the Pareto front, or *non-supported efficient solutions*, which correspond to points lying in the interior of the convex hull of the Pareto front. For BSSFP, identifying even an efficient solution is computationally challenging due to the problem's NP-hardness.

Our work focuses on the case where G is a tree. To the best of our knowledge, no efficient results are known related to the enumeration of efficient solutions of constrained SSF problems involving both weight and center constraints, even for restricted graph classes. The difficulty lies in the exponential growth of the solution space, which arises once a second objective is considered. To address this, we adopt the classical two-phase approach, a general framework for solving bi-objective combinatorial optimization problems [14]. Phase I enumerates all supported efficient solutions, which are obtained through weighted-sum scalarization [13], while Phase II extends the search to non-supported efficient solutions. Both phases reduce to solving single-objective subproblems on trees, which motivates the design of an efficient dynamic programming algorithm. Our main contribution lies in presenting that these subproblems can be solved exactly in polynomial time by a dynamic programming scheme inspired by C. T. Nguyen et al. [10] and V. H. Nguyen [12], but for the minimum version instead of the maximum one. A key idea is the use of Lagrangian relaxation to address the possible cases of the children in the tree. This relaxation allows each child to be considered independently, so that the cases of all children can then be combined systematically into the dynamic programming recurrence. As a result, we obtain a bottom-up recursion that can solve both scalarized formulations for Phase I and ϵ-constraint formulations for Phase II.

In this paper, we consider an edge-weighted tree $T = (V, E, w)$. In combinatorial optimization, it is natural to consider the weight values w as positive integers. The objectives of the BSSFP are to minimize both the total weight and the number of centers, which can be formulated as

$$\min_{x \in \mathcal{X}} (f_1(x), f_2(x)),$$

where $f_1(x)$ represents the total weight, $f_2(x)$ represents the number of centers in the star forest, and $\mathcal{X}$ denotes the set of all feasible decision vectors associated with the total weight and the number of centers in spanning star forests in T. Since w is a positive integer vector, the value of the objectives $f_1(x)$ and $f_2(x)$ is a non-negative integer. Let $(f_1, f_2) = (f_1(x), f_2(x))$ denote the objective values corresponding to a decision vector. Let $\mathcal{F}$ represent the set of pairs (f_1, f_2) corresponding to all feasible decision vectors. This work characterizes the feasible solutions of the BSSFP by their objective pairs (f_1, f_2) instead of explicitly listing the decision vectors. Accordingly, two feasible solutions that have the same objective pair (f_1, f_2) are considered equivalent.

The remainder of this chapter is organized as follows. Section 2 introduces Phase I, which enumerates the supported efficient solutions of the BSSFP on trees. By Geoffrion's theorem [7], every optimal solution of a weighted-sum problem with positive weights is a supported efficient solution. To obtain these solutions, we design dynamic programming algorithms for the weighted sum subproblems using Lagrangian relaxation. Section 3 deals with Phase II, which enumerates the non-supported efficient solutions. In the bi-objective problem, each pair of consecutive supported solutions defines a region in the objective space that may contain non-supported solutions. We explore these regions using the ϵ-constraint approach [9], and the resulting subproblems are solved by dynamic programming algorithms specifically designed to address the constraint on the number of centers. Section 4 concludes the chapter and discusses possible directions for future research.

2 Phase I: Determination of supported efficient solutions

This section aims to identify all supported efficient solutions of BSSFP on trees. We first recall that a solution is supported if it is an optimal solution of a weighted sum scalarization of the total edge weight and the number of centers [13]. Given a tree $T = (V, E)$ with $|V| = n$, edge weights $w_{ij} \geq 0$ for all $ij \in E$, and a scalarization coefficient $\alpha \in [0, 1]$, the weighted sum problem is formally written as

$$\mathcal{W}_\alpha = \min_{(f_1, f_2) \in \mathcal{F}} (1 - \alpha) f_1 + \alpha f_2.$$

We now present Procedure FIND_SUPPORTED(), an iterative algorithm for identifying the supported solutions of BSSFP, following the two-phase framework in [6]. The idea is to detect all values of α at which the optimal solution changes. We begin with the two extremes by solving $\mathcal{W}_0 = \min\ f_1$ and $\mathcal{W}_1 = \min\ f_2$, which return the endpoint solutions (f_1^0, f_2^0) and (f_1^1, f_2^1), respectively. Note that a solution where every node is isolated already forms a spanning star forest. Therefore, minimizing f_1 leads to the trivial solution $(f_1^0, f_2^0) = (0, |V|)$. Since (f_1^0, f_2^0) and (f_1^1, f_2^1) are two efficient solutions, it follows that $f_1^0 < f_1^1$ and $f_2^0 > f_2^1$. We keep these endpoints sorted by increasing f_1, and we place the segment between them into a working list of segments. While there is a segment $[(f_1^i, f_2^i), (f_1^j, f_2^j)]$, we test whether there is a new supported solution strictly between them. This solution is associated with the weight α, where the two endpoints give the same weighted sum. Solving $(1-\alpha) f_1^i + \alpha f_2^i = (1-\alpha) f_1^j + \alpha f_2^j$ gives $\alpha = \frac{f_1^j - f_1^i}{(f_1^j - f_1^i) + (f_2^i - f_2^j)}$. We then solve $\mathcal{W}_\alpha$ and obtain a solution (f_1^m, f_2^m). If it is the same as one endpoint, then no new point lies inside the segment, and we close it. If it is different from both endpoints, we have found a new supported solution and insert this point between the two endpoints in the ordered list. Then we push the two new subsegments onto the working list to further subdivide them in the same way. The loop ends when the working list is empty.

Algorithm 1 begins by solving $\mathcal{W}_0$ and $\mathcal{W}_1$ to obtain the two extreme supported solutions (f_1^0, f_2^0) and (f_1^1, f_2^1), respectively. Line 3 initializes the supported efficient solution set $\mathcal{S}$ with these two points and initializes the list L with the segment between them. The loop then iteratively takes one segment: if the two endpoints share the same value in at least one objective, the segment is skipped; otherwise, the scalarization coefficient α is computed and $\mathcal{W}\alpha$ is solved to generate a candidate solution. In Line 10, if this point shares the same value in at least one objective with either endpoint, nothing is added, but if it is distinct, it is inserted into $\mathcal{S}$ and the two resulting subsegments are added to L. The process repeats until no segments remain, at which point the algorithm returns the complete set $\mathcal{S}$ of supported efficient solutions.

We now propose a dynamic programming algorithm that solves $\mathcal{W}_\alpha$ exactly in linear time for trees. Our approach is inspired by prior works on the Maximum Weight Star Forest Problem (MWSFP). C. T. Nguyen et al. [10] designed a dynamic programming algorithm for MWSFP on trees using bottom-up computations on a rooted tree. Then, V. H. Nguyen [12] extended this method to cactus

Algorithm 1 Enumerating supported efficient solutions via $\mathcal{W}_\alpha$

```
Input: A weighted tree $T = (V, E, w)$.
Output: Ordered set $\mathcal{S}$ of supported efficient solutions.
 1: procedure FIND_SUPPORTED()
 2:     solve $\mathcal{W}_0$ to get $(f_1^0, f_2^0)$;    solve $\mathcal{W}_1$ to get $(f_1^1, f_2^1)$
 3:     $\mathcal{S} \leftarrow \{(f_1^0, f_2^0), (f_1^1, f_2^1)\}$ ordered by $f_1$,    $L \leftarrow \{[((f_1^0, f_2^0), (f_1^1, f_2^1)]\}$
 4:     while $L \neq \emptyset$ do
 5:         select and remove $[(f_1^i, f_2^i), (f_1^j, f_2^j)] \in L$
 6:         if min $\{|f_1^j - f_1^i|, |f_2^i - f_2^j|\} \leq 0$ then go to line 4
 7:         end if
 8:         $\alpha = \frac{f_1^j - f_1^i}{(f_1^j - f_1^i) + (f_2^i - f_2^j)}$
 9:         solve $\mathcal{W}_\alpha$ to get $(f_1^m, f_2^m)$
10:         if min $\{|f_1^m - f_1^i|, |f_2^m - f_2^i|, |f_1^m - f_1^j|, |f_2^m - f_2^j|\} \leq 0$ then go to line 4
11:         else
12:             insert $(f_1^m, f_2^m)$ between endpoints in $\mathcal{S}$
13:             $L \leftarrow L \cup \{[(f_1^i, f_2^i), (f_1^m, f_2^m)], [(f_1^m, f_2^m), (f_1^j, f_2^j)]\}$
14:         end if
15:     end while
16:     return $\mathcal{S}$
17: end procedure
```

graphs, in which any two simple cycles share at most one vertex, by recursively processing blocks with tailored dynamic programming for trees and cycles while preserving linear complexity. However, in our case, we deal with the minimum version instead of the maximum one. We thus employ Lagrangian relaxation to address our problem. The recurrence formulas of our dynamic programming algorithm are given in the following lemma.

Lemma 1. *For each node $u \in V$, let $C(u)$ denote the set of children of u, and let $T(u)$ denote the subtree rooted at u. We then define*

- *Center case $\Phi(u)$: value of $\mathcal{W}_\alpha$ of spanning star forests in $T(u)$, given that u is the center of a star with at least one child.*
- *Leaf case $\Psi(u)$: value of $\mathcal{W}_\alpha$ of spanning star forests in $T(u)$, given that u is a leaf in a star.*
- *Isolated case $\Omega(u)$: value of $\mathcal{W}_\alpha$ of spanning star forests in $T(u)$, given that u is isolated.*

Then the following recurrence holds

$$\Phi(u) = \alpha + \sum_{v \in C(u)} \sigma(v) + \min_{v \in C(u)} \left(\Omega(v) + (1-\alpha) w_{uv} - \sigma(v) \right),$$

$$\Psi(u) = \min_{v \in C(u)} \Big[(1-\alpha) w_{uv} + \min\{\Phi(v), \Omega(v)\} + \sum_{x \in C(u) \setminus \{v\}} \sigma(v) \Big],$$

$$\Omega(u) = \alpha + \sum_{v \in C(u)} \sigma(v)\},$$

where for each $v \in C(u), \sigma(v) = \min\{\Phi(v), \Psi(v), \Omega(v)\}$.

Proof. For each $v \in C(u)$, we define the terms $A(v), B(v)$ as follows

$$A(v) = \min\{\Phi(v), \Psi(v), \Omega(v)\}, \text{ and } B(v) = \Omega(v) + (1-\alpha)w_{uv},$$

where $A(v)$ denotes the optimal value of $\mathcal{W}_\alpha$ for $T(v)$ when the edge uv is not used, and $B(v)$ denotes the value when uv is used. Moreover, we define $\sigma(v) = \min\{A(v), B(v)\}$ which represents the optimal value of the subtree $T(v)$ to the value of $\mathcal{W}_\alpha$ for $T(u)$, depending on whether the edge uv is selected. Since $w_{uv} \geq 0$ and $0 \leq \alpha \leq 1$, we have $B(v) \geq \Omega(v) \geq A(v)$, therefore, in our setting $\sigma(v) = A(v) = \min\{\Phi(v), \Psi(v), \Omega(v)\}$. We consider three possible roles of u.

- *Center case* $\Phi(u)$. For each $v \in C(u)$ decide whether to attach v to u, let $\xi_{uv} \in \{0,1\}$ be the binary variable associated to the occurrence of edge uv in the star centered at u, subject to $\sum_{v \in C(u)} \xi_{uv} \geq 1$. The value of $\mathcal{W}_\alpha$ on the subtree $T(u)$ in this case is

$$\alpha + \sum_{v \in C(u)} (A(v) + (B(v) - A(v))\,\xi_{uv}).$$

Then $\Phi(u)$ can be obtained by solving the following optimization problem

$$(\mathcal{W}(u)) \quad \min \left\{\alpha + \sum_{v \in C(u)} \big(A(v) + (B(v) - A(v))\,\xi_{uv}\big)\right\}$$
$$\text{s.t.} \sum_{v \in C(u)} \xi_{uv} \geq 1.$$

Relax $\sum_v \xi_{uv} \geq 1$ with multiplier $\lambda \geq 0$. Let $\mathcal{L}(\xi, \lambda)$ be the Lagrangian of problem $(\mathcal{W}(u))$, then

$$\mathcal{L}(\xi, \lambda) = \alpha + \sum_v A(v) + \lambda + \sum_v \big((B(v) - A(v)) - \lambda\big)\,\xi_{uv}.$$

Moreover, since $\xi_{uv} \in \{0,1\}$, we get that for each $v \in C(u)$

$$\min_{\xi_{uv} \in \{0,1\}} \big((B(v) - A(v)) - \lambda\big)\xi_{uv} = \min\{0, (B(v) - A(v)) - \lambda\}.$$

Hence, the dual problem of problem $(\mathcal{W}(u))$ is

$$\max_{\lambda \geq 0} \quad g(\lambda) = \alpha + \sum_v A(v) + \lambda + \sum_v \min\{0, (B(v) - A(v)) - \lambda\}.$$

An optimal solution $\lambda^\star$ of this dual problem is $\lambda^* = 0$ if there exists v with $B(v) \leq A(v)$, and otherwise $\lambda^\star = \min_v (B(v) - A(v))$. Since the feasible set

$\{\xi \in [0,1]^{C(u)} : \sum_v \xi_{uv} \geq 1\}$ has integral extreme points, the relaxation is tight and the optimal value of this dual problem equals $\Phi(u)$. Hence, we get

$$\begin{aligned}\Phi(u) &= g(\lambda^*)\\ &= \begin{cases}\alpha + \sum_v \min\{A(v), B(v)\}, & \text{if } \exists\, v \text{ with } B(v) \leq A(v),\\ \alpha + \sum_v A(v) + \min_v (B(v) - A(v)), & \text{if } B(v) > A(v) \text{ for all } v.\end{cases}\\ &= \alpha + \sum_{v \in C(u)} \sigma(v) + \min_{v \in C(u)} \big[\Omega(v) + (1-\alpha) w_{uv} - \sigma(v)\big].\end{aligned}$$

- *Leaf case* $\Psi(u)$. The node u must attach to exactly one child. Let $\eta_{uv} \in \{0,1\}$ be the binary variable associated to the occurrence of edge uv when u is a leaf attached to v subject to $\sum_{v \in C(u)} \eta_{uv} = 1$. For every $v \in C(u)$, we denote

$$\Delta(v) = (1-\alpha) w_{uv} + \min\{\Phi(v), \Omega(v)\} - A(v).$$

Then $\Psi(u)$ can be obtained by solving the following optimization problem

$$\min_{\eta \in \{0,1\}^{C(u)}} \sum_{x \in C(u)} A(x) + \sum_{v \in C(u)} \Delta(v)\, \eta_{uv} \quad \text{s.t.} \quad \sum_{v \in C(u)} \eta_{uv} = 1.$$

Relax $\sum_v \eta_{uv} = 1$ with multiplier $\mu \in \mathbb{R}$. Let $\mathcal{L}(\eta, \mu)$ be the Lagrangian of problem $(\mathcal{W}(u))$, then

$$\mathcal{L}(\eta, \mu) = \sum_{x \in C(u)} A(x) + \mu + \sum_{v \in C(u)} \big(\Delta(v) - \mu\big)\, \eta_{uv}.$$

Note that for each v, we have $\min_{\eta_{uv} \in \{0,1\}} (\Delta(v) - \mu)\, \eta_{uv} = \min\{0, \Delta(v) - \mu\}$. Thus, the dual problem of $(\mathcal{W}(u))$ is

$$\max \quad h(\mu) = \sum_{x \in C(u)} A(x) + \mu + \sum_{v \in C(u)} \min\{0, \Delta(v) - \mu\}.$$

The maximum of $h(\mu)$ is reached at $\mu^* = \min_v \Delta(v)$. Since the feasible set $\{\eta \in [0,1]^{C(u)} : \sum_v \eta_{uv} = 1\}$ has integral extreme points, the optimal value of this dual problem equals $\Phi(u)$. Therefore, we have

$$\begin{aligned}\Psi(u) &= h(\mu^*)\\ &= \sum_{x \in C(u)} A(x) + \min_{v \in C(u)} \Delta(v)\\ &= \sum_{x \in C(u)} A(x) + \min_{v \in C(u)} \Big[(1-\alpha) w_{uv} + \min\{\Phi(v), \Omega(v)\} - A(v)\Big]\\ &= \min_{v \in C(u)} \Big[(1-\alpha) w_{uv} + \min\{\Phi(v), \Omega(v)\} + \sum_{x \in C(u) \setminus \{v\}} A(x)\Big].\end{aligned}$$

- *Isolated case* $\Omega(u)$. The node u forms a single-vertex star, hence

$$\Omega(u) = \alpha + \sum_{v\in C(u)} A(v) = \alpha + \sum_{v\in C(u)} \min\{\Phi(v), \Psi(v), \Omega(v)\}.$$

Therefore, the stated recurrence holds for all nodes $u \in V$. □

We now state Procedure 2 to determine the solution of the subproblem $\mathcal{W}_\alpha$. This procedure traverses the tree in post-order so that each node is processed only after all of its children. Lines 2-4 address initialization: if a node is a leaf, the algorithm assigns the base values $(\Phi(u), \Psi(u), \Omega(u)) = (+\infty, 0, \alpha)$. Otherwise, the set of children $C(u)$ is identified for the recursive step. Line 5 records the children of u, and Lines 6-11 compute $(\Phi(u), \Psi(u), \Omega(u))$ according to the recurrence given in Lemma 1. Finally, the algorithm returns the optimal solution at the root by taking the minimum among these three values.

Procedure 2 Algorithm for solving $\mathcal{W}_\alpha$

Input: A tree $T = (V, E)$ with edge weights w_{uv}, a root r and a coefficient $\alpha \in [0, 1]$.
Output: A spanning star forest that minimizes the weighted sum of the total weight and the number of centers, with α as the scalarization coefficient.

```
1: for all nodes $u \in V$ in post-order traversal do
2:     if $C(u) = \varnothing$ then
3:         $(\Phi(u), \Psi(u), \Omega(u)) \leftarrow (+\infty,\ 0,\ \alpha)$
4:     end if
5:     $C(u) \leftarrow$ the set of children of $u$
6:     for all $v \in C(u)$ do
7:         $\sigma(v) \leftarrow \min\{\Phi(v), \Psi(v), \Omega(v) + (1-\alpha)w_{uv}\}$
8:     end for
9:     $\Phi(u) \leftarrow \alpha + \sum_{v\in C(u)} \sigma(v) + \min_{v\in C(u)} \big(\Omega(v) + (1-\alpha)w_{uv} - \sigma(v)\big)$
10:    $\Psi(u) \leftarrow \min_{v\in C(u)} [(1-\alpha)w_{uv} + \min\{\Phi(v), \Omega(v)\} + \sum_{x\in C(u), x\neq v} \min\{\Phi(x), \Psi(x), \Omega(x)\}]$
11:    $\Omega(u) \leftarrow \alpha + \sum_{v\in C(u)} \min\{\Phi(v), \Psi(v), \Omega(v)\}$
12: end for
13: return $\min\{\Phi(r), \Psi(r), \Omega(r)\}$, where $r$ is the root of the tree
```

Theorem 1. *Procedure 2 solves the problem $\mathcal{W}_\alpha$ in $\mathcal{O}(n)$ time, where n is the number of nodes in the tree.*

Proof. The algorithm performs a single post-order traversal of T, computing $\Phi(u)$, $\Psi(u)$, and $\Omega(u)$ for every $u \in V$ using the recurrences of Lemma 1. For a fixed u, all quantities required by these recurrences, including $\sum_{v\in C(u)} \sigma(v)$, the extremal term in $\Phi(u)$, and the unrestricted sums in $\Psi(u)$ and $\Omega(u)$, are obtained in one pass over $C(u)$, requiring $\mathcal{O}(|C(u)|)$ time. Since $\sum_{u\in V} |C(u)| = n-1$, the overall running time is $\mathcal{O}(n)$, and the optimal objective value is obtained by a constant-time evaluation of $\min\{\Phi(r), \Psi(r), \Omega(r)\}$ at the root r. □

3 Phase II: Determination of non-supported efficient solutions

Phase II determines the set of efficient solutions, which are in the triangle defined by two consecutive supported solutions (f_1^i, f_2^i) and (f_1^j, f_2^j), where we assume the solutions are ordered by increasing f_2 so that $f_2^i < f_2^j$. In order to do this, we first recall that an efficient solution is an optimal solution of a ϵ-constraint problem [9]. For the BSSFP, this problem focuses on minimizing the total weight of a spanning star forest in the tree $T = (V, E)$ with edge weights w, subject to a bound on the number of centers k, i.e.,

$$(\mathcal{P}_k) \quad \min \{f_1 \ : \ (f_1, f_2) \in \mathcal{F}, f_2 \leq k\}$$

Algorithm 3 presents Procedure FIND_NONSUPPORTED(), which applies the ϵ-constraint method [9] to enumerate the non-supported efficient solutions. Starting from two consecutive supported solutions (f_1^i, f_2^i) and (f_1^j, f_2^j), the procedure initializes $k = f_2^j - 1$ and sets $U = \emptyset$ as the set of non-supported solutions. At each step, problem $(\mathcal{P}_k)$ is solved to obtain a solution (f_1^u, f_2^u). If this solution is distinct from the endpoints, it is recorded as a non-supported efficient solution and added to the set U. After that, set $k = f_2^u - 1$ and continue the search. The loop continues until no integer k remains strictly between f_2^i and f_2^j, at which point all non-supported solutions in that range have been found.

Algorithm 3 Enumerating non-supported efficient solutions via $(\mathcal{P}_k)$

Input: A weighted tree $T = (V, E, w)$ and two consecutive supported efficient solution $(f_1^i, f_2^i), (f_1^j, f_2^j)$ with $f_2^i < f_2^j$ of BSSFP for T.
Output: Set U of non-supported efficient solutions between $(f_1^i, f_2^i), (f_1^j, f_2^j)$.

```
1: procedure FIND_NONSUPPORTED()
2:     $U = \emptyset, k = f_2^j - 1$
3:     while $k \geq f_2^i + 1$ do
4:         solve $(\mathcal{P}_k)$ to obtain solution $(f_1^u, f_2^u)$
5:         add $(f_1^u, f_2^u)$ to $U$ and set $k = f_2^u - 1$
6:     end while
7: end procedure
```

We then present a dynamic programming algorithm to solve $(\mathcal{P}_k)$, which minimizes the total weight of a spanning star forest in a rooted tree $T = (V, E)$ with edge weights w_{uv} with the constraint that there are at most k centers. The algorithm also uses bottom-up dynamic programming with post-order traversal. The recurrence relations are stated in the following lemma.

Lemma 2. *For each node $u \in V$ and $\ell \in \{0, 1, \ldots, k\}$, let $C(u)$ denote the set of children of u, and let $T(u)$ denote the subtree rooted at u, we define*

- *Isolated case $\Omega(u, \ell)$: minimum weight (i.e., objective value of $(\mathcal{P}_k)$) in $T(u)$ with u isolated and exactly ℓ centers.*

- *Leaf case $\Psi(u,\ell)$: minimum weight of spanning star forests in $T(u)$ with u as a leaf in the star and exactly ℓ centers.*
- *Center case $\Phi(u,\ell)$: minimum weight of spanning star forests in $T(u)$ with u as a star center (at least one child) and exactly ℓ centers.*

For each $\ell \in \{0,1,\dots,k\}$, let $\sigma(v,\ell) = \min\{\Phi(v,\ell),\Psi(v,\ell),\Omega(v,\ell)\}$. Then the following recurrences hold

$$\begin{aligned}
\Phi(u,\ell) &= \min_{\substack{S\subseteq C(u),\ S\neq\varnothing \\ \sum_{x\in C(u)} \ell_x = \ell-1+|S|}} \Big[\sum_{v\in S}\big(w_{uv}+\Omega(v,\ell_v)\big) + \sum_{x\in C(u)\setminus S}\sigma(x,\ell_x)\Big],\\
\Psi(u,\ell) &= \min_{\substack{v\in C(u),\ \{\ell_x\}_{x\in C(u)} \\ \sum_{x\in C(u)} \ell_x = \ell}} \Big[w_{uv} + \min\{\Phi(v,\ell_v),\Omega(v,\ell_v)\} \\
&\qquad + \sum_{x\in C(u), x\neq v}\sigma(x,\ell_x)\Big],\\
\Omega(u,\ell) &= \min_{\substack{\{\ell_v\}_{v\in C(u)} \\ \sum_{v\in C(u)} \ell_v = \ell-1}} \sum_{v\in C(u)}\sigma(v,\ell_v).
\end{aligned}$$

Proof. For each $v \in C(u)$ and $t \in \{0,\dots,k\}$, we let

$$\sigma(v,t) = \min\{\Phi(v,t),\Psi(v,t),\Omega(v,t)\},$$

which is the optimal total weight in $T(v)$ when the edge uv does not occur in the component containing u.

- *Center case $\Phi(u,\ell)$.* Let $\xi_{uv} \in \{0,1\}$ be the binary variable associated to the occurrence of edge uv in the star centered at u, with $\sum_v \xi_{uv} \geq 1$, and integers $\ell_v \geq 0$ for the number of centers used in $T(v)$. Then $\Phi(u,l)$ can be obtained by solving the following optimization problem

$$\begin{aligned}
\min_{\xi,\ell.} \ & \sum_{v:\xi_{uv}=1}\big(w_{uv}+\Omega(v,\ell_v)\big) + \sum_{x:\xi_{ux}=0}\sigma(x,\ell_x) \qquad (1)\\
\text{s.t.} \ & \sum_v \xi_{uv} \geq 1, \qquad \sum_{x\in C(u)} \ell_x - |\{v:\xi_{uv}=1\}| = \ell - 1.
\end{aligned}$$

Relax the two constraints with multipliers $\lambda \geq 0$ and $\gamma \in \mathbb{R}$. The Lagrangian of (1) is

$$\begin{aligned}
\mathcal{L}(\xi,\ell;\lambda,\gamma) = & \sum_v \Big((1-\xi_{uv})\sigma(v,\ell_v) + \xi_{uv}\big(w_{uv}+\Omega(v,\ell_v)\big)\Big)\\
& + \lambda\Big(1-\sum_v \xi_{uv}\Big) + \gamma\Big((\ell-1) - \sum_v \ell_v + \sum_v \xi_{uv}\Big).
\end{aligned}$$

For each v, let $A_\gamma(v) = \min_{t\geq 0}\{\sigma(v,t)-\gamma t\}$, and $B_\gamma(v) = \min_{t\geq 0}\{\Omega(v,t) - \gamma t\} + (\gamma + w_{uv})$.

Then minimizing $\mathcal{L}(\xi, \ell; \lambda, \gamma)$ over ℓ_v, we get the dual problem $\max_{\lambda \geq 0,\ \gamma \in \mathbb{R}} g(\lambda, \gamma)$ of (1), where

$$\begin{aligned} g(\lambda,\gamma) = \inf_{\xi,\ell} \mathcal{L} &= \lambda + \gamma(\ell - 1) + \sum_v \min\{A_\gamma(v),\ B_\gamma(v) - \lambda\} \\ &= \gamma(\ell-1) + \sum_v \min\{A_\gamma(v), B_\gamma(v)\} \\ &\quad + \max_{\lambda \geq 0} \Big[\lambda + \sum_v \min\{0,\ \varDelta_\gamma(v) - \lambda\}\Big], \end{aligned}$$

with $\varDelta_\gamma(v) = B_\gamma(v) - A_\gamma(v)$. Maximizing $g(\lambda, \gamma)$ over $\lambda \geq 0$ gives us an optimal solution for λ

$$\lambda^\star = \begin{cases} 0, & \text{if there exists } v \text{ with } B_\gamma(v) \leq A_\gamma(v), \\ \min_{v \in C(u)} \varDelta_\gamma(v), & \text{otherwise.} \end{cases}$$

Then we get the optimal value of $g(\lambda, \gamma)$ as follows

$$\begin{aligned} \max_{\gamma \in \mathbb{R}}\ g(\lambda,\gamma) = \gamma(\ell-1) &+ \sum_{v \in C(u)} \min\{A_\gamma(v), B_\gamma(v)\} \\ &+ \min_{v \in C(u)} \Big(B_\gamma(v) - \min\{A_\gamma(v), B_\gamma(v)\}\Big). \end{aligned}$$

We then maximize $g(\lambda, \gamma)$ over $\gamma \in \mathbb{R}$. Note that the optimal value of this dual problem equals $\varPhi(u)$. Thus, we have

$$\begin{aligned} \varPhi(u,\ell) &= \max_{\lambda \geq 0,\ \gamma \in \mathbb{R}} g(\lambda,\gamma) \\ &= \min_{S \subseteq C(u),\ S \neq \varnothing}\ \max_{\gamma \in \mathbb{R}} \Big\{ \gamma(\ell - 1 + |S|) + \sum_{v \in S} \min_{t \geq 0} \big(\varOmega(v,t) - \gamma t\big) \\ &\qquad + \sum_{x \in C(u) \setminus S} \min_{t \geq 0} \big(\sigma(x,t) - \gamma t\big) \Big\} \\ &= \min_{\substack{S \subseteq C(u),\ S \neq \varnothing \\ \sum_{x \in C(u)} \ell_x = \ell - 1 + |S|}} \Big[\sum_{v \in S} \big(w_{uv} + \varOmega(v, \ell_v)\big) + \sum_{x \in C(u) \setminus S} \sigma(x, \ell_x) \Big]. \end{aligned}$$

- *Leaf case $\varPsi(u,\ell)$ and isolated case $\varOmega(u)$.* These two cases correspond respectively to the situation where vertex u is a leaf or an isolated vertex in the tree. Their recursive formulations are obtained by applying the same decomposition principle as for the case $\varPhi(u,\ell)$. The complete proofs are provided in [11]. □

From the recurrence formulation in Lemma 2, we obtain a dynamic programming algorithm to solve each problem $(\mathcal{P}_k)$. The overall complexity of solving the BSSFP on trees is given by the combination of Phase I and Phase II, as stated in the following theorem.

Theorem 2. *The efficient solution set of BSSFP on trees can be determined in $O(n^3)$ time and $O(n^2)$ space.*

Proof. To show the time and space complexity of our method for solving BSSFP on trees, we only need to consider the complexity of Phase II. We first show that the number of problems $(\mathcal{P}_k)$ that have to be solved is equal to the number $|U|$ of non-supported efficient solutions. Considering two consecutive supported efficient solutions $(f_1^i, f_2^i), (f_1^j, f_2^j)$ with $f_2^i < f_2^j$. We start with $k = f_2^j - 1$ and solve $(\mathcal{P}_k)$. Note that the endpoints are consecutive supported solutions; there is no supported efficient solution (f_1, f_2) with $f_2 \in (f_2^i, f_2^j)$. Hence, every executed call returns a rightmost non-supported efficient solution (f_1^m, f_2^m) with $f_2^i < f_2^m \leq k$. Record (f_1^m, f_2^m) and split the interval into (f_2^i, f_2^m) and (f_2^m, f_2^j), on each subinterval apply the same rule. By induction on the number $|U_{ij}|$ of non-supported solutions inside (f_2^i, f_2^j), the step case removes one point and recurses on the two subintervals, the number of $(\mathcal{P}_k)$ solves on this interval is $|U_{ij}|$. Summing over all intervals, we get that the number of iterations for solving $(\mathcal{P}_k)$ in phase II is $|U|$. Since each efficient solution corresponds to a distinct value of f_2 in $\{0, \ldots, n\}$, it follows that $|U| < n$.

We now build a memoization table $\sigma(v, \ell)$ for every node $v \in V$ and every $\ell \in \{0, \ldots, n\}$. All values are computed bottom-up on the rooted tree using the recurrences of Lemma 2. At each node, combining the contributions of the children requires distributing ℓ among their subtrees, which can be done in $O(\deg(v)\, n^2)$ time. Summing $\deg(v)$ over all nodes gives a total running time of $O(n^3)$, and the size of the table is $O(n^2)$. Once the table is available, each problem $(\mathcal{P}_k)$ can be solved in constant time by reading the corresponding entry. Since Phase II needs to evaluate exactly $|U|$ such problems, the additional time is $O(|U|)$. Therefore, the full efficient solution set is obtained in $O(n^3)$ time using $O(n^2)$ space. □

4 Conclusion

This work has introduced and studied the Bi-objective Spanning Star Forest Problem (BSSFP), where the two objectives are to minimize both the number of centers and the total edge weight. Our focus has been on the case where the input is a tree. To enumerate the efficient solution set for BSSFP, we presented a polynomial-time algorithm through a two-phase dynamic programming approach. The key idea is the use of a Lagrangian relaxation to justify subtree separability, which ensures that the recurrence relations are exact. Based on the results obtained for trees, it is natural to consider extending the algorithm in this paper to graphs that can be reduced to trees, such as cactus graphs. Another promising line of research is to investigate approximation algorithms for general graphs by comparing the solutions on a general graph with those obtained from its spanning trees.

References

1. A. Agra, D. Cardoso, O. Cerdeira, and E. Rocha. A spanning star forest model for the diversity problem in the automobile industry. In: *ECCO XVIII, Minsk*, 2005.
2. M. Aïder, L. Aoudia, M. Baiou, A. R. Mahjoub, and V. H. Nguyen. On the star forest polytope for trees and cycles. *RAIRO-Operations Research*, 53, 2019.
3. A. Andersson. General Balanced Trees. *Journal of Algorithms*, 30(1):1–18, 1999.
4. S. Athanassopoulos, I. Caragiannis, C. Kaklamanis, and M. Kyropoulou. An improved approximation bound for spanning star forest and color saving. In: *Mathematical Foundations of Computer Science (MFCS)*, pp. 90–101, 2009.
5. O. Briant and D. Naddef. The Optimal Diversity Management Problem. *Operations Research*, 52(4):515–526, 2004.
6. C. Filippi and G. Stevanato. A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits. *4OR*, 11(3):253–276, 2013.
7. A. M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22(3):618–630, 1968.
8. O. Kariv and L. Hakimi. An algorithmic approach to network location problems. II: The p-medians. *Operations Research*, 37(3):539–560, 1979.
9. G. Mavrotas. Effective implementation of the epsilon-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213:455–465, 2009.
10. C. T. Nguyen, J. Shen, M. Hou, L. Sheng, W. Miller, and L. Zhang. Approximating the spanning star forest problem and its applications to genomic sequence alignment. *SIAM Journal on Computing*, pp. 946–962, 2008.
11. T.L. Nguyen, V.H. Nguyen. Polynomial time algorithm for a Bi-objective Spanning Star Forest Problem on Trees. 2025. URL https://hal.science/hal-05284724
12. V. H. Nguyen. The maximum weight spanning star forest problem on cactus graphs. *Discrete Mathematics, Algorithms and Applications*, World Scientific Publishing, 2015.
13. O. Özpeynirci and M. Köksalan. An exact algorithm for finding extreme supported non-dominated points of multiobjective mixed integer problems. *Management Science*, 56(12):2302–2315, 2010.
14. E. L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20:149–165, 1995.