Minimum Robinson-Foulds Distance Supertree

Xilin Yu, Thien Le, Sarah Christensen, Erin Molloy, Tandy Warnow

June 4, 2019

1 Introduction

2 The Maximum Bipartition Support Supertree Problem

2.1 Terminology and Preliminary

Throughout the paper, we consider only unrooted trees. For any tree T, let $V(T)$, $E(T)$, and $L(T)$ denote the vertex set, the edge set, and the leaf set of T, respectively. For any $v \in V(T)$, let $N_T(v)$ A tree is *fully resolved* if every non-leaf node has degree 3. Let $\mathcal{T}_S$ denote the set of all fully resolved trees on leaf set S. In any tree T, each edge e induces a bipartition $\pi_e := A|B$ of the leaf set, where A and B are the leaves in the two components of $T - e$, respectively. A bipartition $A|B$ is non-trivial if both sides have size at least 2. For a tree T, $C(T) := \{\pi_e \mid e \in E(T)\}$ denotes the set of all bipartitions of T. For a fully resolved tree with n leaves, $C(T)$ contains $2n-3$ bipartitions, exactly $n-3$ of which are non-trivial. A tree T' is a *refinement* of T if T can be obtained from T' by contracting a set of edges. Equivalently, T' is a refinement of T if and only if $C(T) \subseteq C(T')$.

Two bipartitions π_1 and π_2 of the same leaf set are *compatible* if and only if there exists a tree T such that $\pi_1, \pi_2 \in C(T)$. The following theorem and corollary give other categorizations of compatibility.

Theorem 1 (Theorem 2.20 of [1])**.** *A pair of bipartitions $A|B$ and $A'|B'$ of the same set is compatible if and only if at least one of the four pairwise intersections $A \cap A'$, $A \cap B'$, $B \cap A'$, $B \cap B'$ is empty.*

Corollary 1. *A pair of bipartitions $A|B$ and $A'|B'$ of the same set is compatible if and only if one side of $A|B$ is a subset of one side of $A'|B'$.*

A tree T restricted to a subset R of its leaf set, denoted $T|_R$, is the minimal subtree of T spanning R with nodes of degree two suppressed. A bipartition $\pi = A|B$ restricted to a subset $R \subseteq A \cup B$ is $\pi|_R = A \cap R|B \cap R$. We have the following intuitive lemma with its proof in the appendix.

Lemma 1. *Let T be a tree with leaf set S and let $\pi = A|B \in C(T)$ be a bipartition induced by $e \in E(T)$. Let $R \subseteq S$.*

1. *If $R \cap A = \emptyset$ or $R \cap B = \emptyset$, then $e \notin E(T|_R)$.*
2. *If $R \cap A \neq \emptyset$ and $R \cap B \neq \emptyset$, then for any $\pi' \in C(T|_R)$ induced by $e' \in E(T|_R)$, $\pi|_R = \pi'$ if and only if $e \in P(e')$.*

Corollary 2. *Let T be a tree with leaf set S and let $\pi = A|B \in C(T)$ be a bipartition induced by $e \in E(T)$. Let $R \subseteq S$ such that $R \cap A \neq \emptyset$ and $R \cap B \neq \emptyset$. Then $\pi|_R \in C(T|_R)$.*

Definition 1. *For two trees T, T' with the same leaf set, the bipartition support of them is* $\mathrm{bisup}(T, T') := |C(T) \cap C(T')|$.

Bipartition support measures the similarity between the topology of the trees.

2.2 Problem Statement

Let T_1 and T_2 be two fully resolved trees on leaf sets S_1 and S_2, respectively, such that $X := S_1 \cap S_2 \neq \emptyset$. Let $S := S_1 \cup S_2$. The maximum bipartition support supertree problem on two input trees, abbreviated Max-Bisup-Supertree-2, finds a fully resolved supertree T^* on leaf set S that maximizes the sum of the bipartition support of T^* with respect to T_1 and T_2. That is,

$$\begin{aligned} T^* &= \underset{T \in \mathcal{T}_S}{\mathrm{argmax}}\, \mathrm{bisup}(T|_{S_1}, T_1) + \mathrm{bisup}(T|_{S_2}, T_2) \\ &= \underset{T \in \mathcal{T}_S}{\mathrm{argmax}}\, |C(T|_{S_1}) \cap C(T_1)| + |C(T|_{S_2}) \cap C(T_2)|. \end{aligned}$$

We call $\mathrm{bisup}(T|_{S_1}, T_1) + \mathrm{bisup}(T|_{S_2}, T_2)$ the support score of T when T_1 and T_2 are clear from context.

The more general maximum bipartition support supertree problem on a set of N input trees, abbreviated Max-Bisup-Supertree-N, takes in a set of input trees $T_1, T_2, \ldots, T_N$ with leaf sets $S_1, S_2, \ldots, S_N$, respectively. Max-Bisup-Supertree-N finds a fully resolved supertree T^* on leaf set S that maximizes the sum of the bipartition support of T^* with respect to every input tree. That is,

$$T^* = \underset{T \in \mathcal{T}_S}{\mathrm{argmax}} \sum_{i \in [N]} \mathrm{bisup}(T|_{S_i}, T_i)$$

2.3 Algorithm

We present a polynomial time algorithm for Max-Bisup-Supertree-2 in this subsection. We first set up the notations for the algorithm and the analysis. Let

T_1, T_2, S_1, S_2, and X be defined as from the problem statement. Let $T_1|_X$ and $T_2|_X$ be the backbone trees of T_1 and T_2, respectively. Let Π be the set of bipartitions of X. Let Triv and NonTriv denotes the set of trivial and non-trivial bipartitions in $C(T_1|_X) \cup C(T_2|_X)$. For each $e \in E(T_i|_X)$, $i \in \{1,2\}$, let $P(e)$ denote the path in T_i from which e is obtained by suppressing all degree-two nodes. Let $w(e)$ be the number of edges on $P(e)$.

We define a weight function $w : \Pi \to \mathbb{N}_{\geq 0}$ such that for any bipartition π of X, $w(\pi) = w(e_1) + w(e_2)$, where e_i induces π in $T_i|_X$ for $i \in \{1,2\}$. If for any $i \in \{1,2\}$, no e_i exists that induces π in $T_i|_X$, then we use $w(e_i) = 0$.

For each $i \in \{1,2\}$ and each $e \in E(T_i|_X)$, let $\mathrm{In}(e)$ be the set of internal nodes of $P(e)$. For each $v \in \mathrm{In}(e)$, let $L(v)$ be the set of leaves in $S_i \setminus X$ whose connecting path to the backbone tree $T_i|_X$ goes through v and let $T(v)$ be the minimal subtree spanning $L(v)$ in T_i. We say $T(v)$ is an extra subtree attached to v. Consider $T(v)$ rooted at the node u which is the neighbor of v in $T(v)$. Let $\mathcal{T}(e) := \{T(v) \mid v \in \mathrm{In}(e)\}$. Then $\mathcal{T}(e)$ is the set of extra subtrees attached to internal nodes of $P(e)$ in T_i. We note that $|\mathcal{T}(e)| = |\mathrm{In}(e)| = w(e) - 1$. For any bipartition $\pi \in C(T_1|_X) \cup C(T_2|_X)$, we denote $\mathcal{T}(\pi) := \mathcal{T}(e_1) \cup \mathcal{T}(e_2)$, where e_i is the edge that induces π in $T_i|_X$ for $i \in \{1,2\}$ if $\pi \in C(T_i|_X)$. Let $\mathrm{Extra}(T_i) := \bigcup_{e \in E(T_i|_X)} \mathcal{T}(e)$. Then $\mathrm{Extra} := \mathrm{Extra}(T_1) \cup \mathrm{Extra}(T_2)$ denotes the set of all extra subtrees in T_1 and T_2. figure to help

For the analysis of the algorithm, we differentiate between two kinds of bipartitions in $C(T_1) \cup C(T_2)$. Let $\Pi_Y = \{\pi = A|B \in C(T_1) \cup C(T_2) \mid \text{either } A \cap X = \emptyset, \text{ or } B \cap X = \emptyset\}$. Let $\Pi_X = \{\pi = A|B \in C(T_1) \cup C(T_2) \mid A \cap X \neq \emptyset \text{ and } B \cap X \neq \emptyset\}$. Intuitively, Π_X is the set of bipartitions in $C(T_1) \cup C(T_2)$ that are induced by edges in the backbone trees $T_1|_X$ and $T_2|_X$ while Π_Y is the set of bipartitions in $C(T_1) \cup C(T_2)$ that are induced by edges inside extra subtrees or connecting extra subtrees to the backbone trees.

Let $p_X(T)$ and $p_Y(T)$ (we omit the parameters T_1 and T_2 for brevity) be the contributions to the support score of T from bipartitions of Π_X and Π_Y for any $T \in \mathcal{T}_S$, respectively. Formally, we have

$$p_X(T) = |C(T|_{S_1}) \cap C(T_1) \cap \Pi_X| + |C(T|_{S_2}) \cap C(T_2) \cap \Pi_X|,$$
$$p_Y(T) = |C(T|_{S_1}) \cap C(T_1) \cap \Pi_Y| + |C(T|_{S_2}) \cap C(T_2) \cap \Pi_Y|.$$

Claim 1. *If Algorithm 1 returns a tree T such that $p_X(T) \geq p_X(T')$ and $p_Y(T) \geq p_Y(T')$ for any tree T' with leaf set S, then Algorithm 1 solves* Max-Bisup-Supertree-2 *correctly.*

Proof. By definition of support score, any bipartition can only contribute to the support score if it is in $C(T_1) \cup C(T_2)$. It follows by definition of Π_X and Π_Y that Π_X and Π_Y is a disjoint decomposition of $C(T_1) \cup C(T_2)$. Thus, the support score of T equals $p_X(T) + p_Y(T)$ for any tree T on leaf set S. Then if $p_X(T) \geq p_X(T')$ and $p_Y(T) \geq p_Y(T')$ for any tree T' with leaf set S, T

Algorithm 1 Max-BiSup Supertree

Input: two fully resolved trees T_1, T_2 with leaf sets S_1 and S_2 where $S_1 \cap S_2 = X \neq \emptyset$

Output: a fully resolved supertree T on leaf set $S = S_1 \cup S_2$ that maximizes the support score

compute $C(T_1|_X)$ and $C(T_2|_X)$

for each $\pi \in C(T_1|_X) \cup C(T_2|_X)$ **do**

  compute $\mathcal{T}(\pi)$ and $w(\pi)$

construct T by having a star of leaf set X with center vertex $\hat{v}$ and connecting the root of each $t \in \text{Extra}$ to $\hat{v}$, let $\hat{T} = T$

for each $\pi \in \text{Triv}$ **do**

  $T \leftarrow \text{Refine-Triv}(T_1, T_2, T, \pi, \hat{v}, \mathcal{T})$

let $\tilde{T} = T$

construct the incompatibility graph $G = (V_1 \cup V_2, E)$, where $V_1 = C(T_1|_X) - C(T_2|_X)$ and $V_2 = C(T_2|_X) - C(T_1|_X)$, and $E = \{(\pi, \pi') \mid \pi \in V_1, \pi' \in V_2$, π is not compatible with $\pi'\}$

compute the maximum weight independent set I in G with weight w

let $H(\hat{v}) = \text{NonTriv} \cap (C(T_1|_X) \cup C(T_2|_X))$

for each $\pi \in \text{NonTriv} \cap (C(T_1|_X) \cup C(T_2|_X))$ **do**

  $sv(\pi) = \hat{v}$

for each $\pi \in \text{NonTriv} \cap (I \cup (C(T_1|_X) \cap C(T_2|_X)))$ **do**

  $T \leftarrow \text{Refine}(T_1, T_2, T, \pi, H, sv, \mathcal{T})$

let $T^* = T$

refine T arbitrarily at polytomies until it is fully resolved

return T

Algorithm 2 Refine-Triv

Input: two trees T_1, T_2 with leaf sets S_1 and S_2 where $S_1 \cap S_2 = X \neq \emptyset$, an unrooted tree T on leaf set $S = S_1 \cup S_2$, a trivial bipartition $\pi = A|b$ of X, a vertex $\hat{v} \in V(T)$, a dictionary $\mathcal{T}$

Output: an tree T' which is a refinement of T such that $\pi \in C(T'|_X)$

detach all extra subtrees in $\mathcal{T}(\pi)$ from $\hat{v}$ and attach them onto $(\hat{v}, b)$ such that the subtrees from $\mathcal{T}(e_1)$ and subtrees from $\mathcal{T}(e_2)$ are side by side and each group respects the ordering of subtrees in T_i

return the resulting tree T'

Algorithm 3 Refine

Input: two trees T_1, T_2 with leaf sets S_1 and S_2 where $S_1 \cap S_2 = X \neq \emptyset$, an unrooted tree T on leaf set $S = S_1 \cup S_2$, a bipartition $\pi = A|B$ of X, a dictionary H, a dictionary sv, a dictionary $\mathcal{T}$

Output: an tree T' which is a refinement of T such that $\pi \in C(T'|_X)$

$v \leftarrow sv(\pi)$
compute $N_A := \{u \in N_T(v) \mid \exists a \in A \text{ such that } u \text{ can reach } a \text{ in } T - v\}$ and $N_B := \{u \in N_T(v) \mid \exists b \in B \text{ such that } u \text{ can reach } b \text{ in } T - v\}$.
$V(T) \leftarrow V(T) \cup \{v_a, v_b\}$, $E(T) \leftarrow E(T) \cup \{(v_a, v_b)\}$
$H(v_a) \leftarrow \emptyset, H(v_b) \leftarrow \emptyset$
for each $u \in N_A \cup N_B$ **do**
    if $u \in N_A$ **then** connect u to v_a
    else connect u to v_b
detach all extra subtrees in $\mathcal{T}(\pi)$ from v and attach them onto (v_a, v_b) such that the subtrees from $\mathcal{T}(e_1)$ and subtrees from $\mathcal{T}(e_2)$ are side by side and each group respects the ordering of subtrees in T_i
for each bipartition $\pi' = A'|B' \in H(v)$ such that $\pi' \neq \pi$ **do**
    detach all extra subtrees in $\mathcal{T}(\pi')$ from v
    if $A' \subseteq A$ or $B' \subseteq A$ **then**
        $sv(\pi') = v_a$ and $H(v_a) \leftarrow H(v_a) + \pi'$
        attach all extra subtrees in $\mathcal{T}(\pi')$ to v_a
    else if $A' \subseteq B$ or $B' \subseteq B$ **then**
        $sv(\pi') = v_b$ and $H(v_b) \leftarrow H(v_b) + \pi'$
        attach all extra subtrees in $\mathcal{T}(\pi')$ to v_b
    else
        discard π' and attach all extra subtrees in $\mathcal{T}(\pi')$ to either v_a or v_b
for each remaining extra subtree attached to v **do**
    detach it from v and attach it to either v_a or v_b
delete v and incident edges from T
return the resulting tree T'

achieves the maximum support score among all trees of leaf set S, in particular, it achieves the maximum support score among all trees in $\mathcal{T}_S$. $\square$

Therefore, it is enough for us to show that Algorithm 1 finds a tree T that maximizes both $p_X(T)$ and $p_Y(T)$ at the same time.

Lemma 2. *For any tree T of leaf set S and any refinement T' of T, $p_X(T') \geq p_X(T)$ and $p_Y(T') \geq p_Y(T)$.*

Proof. Since T' is an refinement of T, $C(T|_{S_i}) \subseteq C(T'|_{S_i})$ for any $i \in \{1,2\}$. Therefore, $|C(T|_{S_i}) \cap C(T_i) \cap \Pi_X| \leq |C(T'|_{S_i}) \cap C(T_i) \cap \Pi_X|$ for any $i \in \{1,2\}$, and thus $p_X(T) \leq p_X(T')$. Similarly, $|C(T|_{S_i}) \cap C(T_i) \cap \Pi_Y| \leq |C(T'|_{S_i}) \cap C(T_i) \cap \Pi_Y|$ for any $i \in \{1,2\}$, and thus $p_Y(T) \leq p_Y(T')$. $\square$

Lemma 3. *For any tree T of leaf set S, $p_Y(T) \leq |\Pi_Y|$. In particular, let $\hat{T}$ be the tree constructed in Algorithm 1. Then, $p_Y(\hat{T}) = |\Pi_Y|$.*

Proof. Since T_1 and T_2 has different leaf sets, $C(T_1)$ and $C(T_2)$ are disjoint. Since $\Pi_Y \subseteq C(T_1) \cup C(T_2)$, $C(T_1) \cap \Pi_Y$ and $C(T_2) \cap \Pi_Y$ forms a disjoint decomposition of Π_Y. By definition of $p_Y(\cdot)$, for any tree T of leaf set S,

$$\begin{aligned} p_Y(T) &= |C(T|_{S_1}) \cap C(T_1) \cap \Pi_Y| + |C(T|_{S_2}) \cap C(T_2) \cap \Pi_Y| \\ &\leq |C(T_1) \cap \Pi_Y| + |C(T_2) \cap \Pi_Y| \\ &= |\Pi_Y|. \end{aligned}$$

Fix any $\pi = A|B \in \Pi_Y$. By definition of Π_Y, either $A \cap X = \emptyset$ or $B \cap X = \emptyset$. Assume without loss of generality that $A \cap X = \emptyset$. If $\pi \in C(T_1)$, let e_1 be the edge that induces π in T_1. Then $A \subseteq S_1 \backslash X$, which implies either e_1 is an internal edge in an extra subtree in $\text{Extra}(T_1)$, or e_1 connects one extra subtree in $\text{Extra}(T_1)$ to the backbone $T_1|_X$. In either case, the construction of $\hat{T}$ ensures that $\pi \in C(\hat{T}|_{S_1})$. Similarly if $\pi \in C(T_2)$, then $\pi \in C(\hat{T}|_{S_2})$ by construction. Therefore, each bipartition $\pi \in \Pi_Y$ contributes 1 to $|C(\hat{T}|_{S_i}) \cap C(T_i) \cap \Pi_Y|$ for exactly one $i \in \{1,2\}$ and thus it contributes 1 to $p_Y(\hat{T})$. Hence, $p_Y(\hat{T}) = |\Pi_Y|$. $\square$

Claim 2. *Let $\hat{T}$ be the tree constructed in Algorithm 1, then $p_X(\hat{T}) = 2|X|$.*

Proof. Let the center of the star from which $\hat{T}$ is constructed be the center of $\hat{T}$. For each $v \in X$, consider the bipartition $\pi_v = \{v\} \mid S\backslash\{v\}$ induced by the edge that connects the leaf v to the center. It is easy to see that $\pi_v|_{S_i} = \{v\} \mid S_i\backslash\{v\} \in C(T_i) \cap C(\hat{T}|_{S_i})$ for any $i \in \{1,2\}$ as $\pi_v|_{S_i}$ is a trivial bipartition of S_i and must be present in any tree on leaf set S_i. We also know $\pi_v|_{S_i} \in \Pi_X$ as $\pi_v \in C(T_1) \cup C(T_2)$ and both sides of π_v has non-empty intersection with X. Thus, $\pi_v|_{S_i} \in C(\hat{T}|_{S_i}) \cap C(T_i) \cap \Pi_X$ for any $i \in \{1,2\}$. So for each $v \in X$, $\pi_v|_{S_1}$ and $\pi_v|_{S_2}$ each contributes 1 to $p_X(\hat{T})$. Therefore, $p_X(\hat{T}) \geq 2|X|$.

Fix any bipartition $\pi = A|B$ induced by any other edge of $\hat{T}$ such that $\pi|_{S_i} \in C(\hat{T}|_{S_i})$ for some $i \in \{1,2\}$. By construction of $\hat{T}$, the edge inducing π is either inside an extra subtree or connecting the root of an extra subtree to the center Therefore, either $A \subseteq S\backslash X$ or $B \subseteq S\backslash X$, which implies $\pi|_{S_i} \notin \Pi_X$ for any $i \in \{1,2\}$. Hence, there is no other bipartition of $\hat{T}$ such that when restrict to S_i contributes to $p_X(\hat{T})$. Therefore, $p_X(\hat{T}) = 2|X|$. $\square$

Lemma 4. *Let $\pi = A|B$ be a bipartition of X. Let T be a tree of leaf set S such that $\pi \notin C(T|_X)$ and all bipartitions in $C(T|_X)$ are compatible with π. Let T' be a refinement of T such that for all $\pi' \in C(T'|_{S_i})\backslash C(T|_{S_i})$ for some $i \in \{1,2\}$, $\pi'|_X = \pi$. Then, $p_X(T') - p_X(T) \leq w(\pi)$.*

Proof. By definition of $p_X(\cdot)$,

$$\begin{aligned}
&p_X(T') - p_X(T) \\
=&|C(T'|_{S_1}) \cap C(T_1) \cap \Pi_X| + |C(T'|_{S_2}) \cap C(T_2) \cap \Pi_X| \\
&- (|C(T|_{S_1}) \cap C(T_1) \cap \Pi_X| + |C(T|_{S_2}) \cap C(T_2) \cap \Pi_X|) \\
=&|(C(T'|_{S_1})\backslash C(T|_{S_1})) \cap C(T_1) \cap \Pi_X| + |(C(T'|_{S_2})\backslash C(T|_{S_2})) \cap C(T_2) \cap \Pi_X| \\
=&\sum_{i=1,2} |(C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X|.
\end{aligned}$$

Therefore, we only need to prove that $\sum_{i=1,2} |(C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X| \leq w(\pi)$. For any $\pi' \in (C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X$ for any $i \in \{1,2\}$, we have $\pi'|_X = \pi$.

We differentiate three different cases for the proof of the above statement: 1) $\pi \notin C(T_1|_X) \cup C(T_2|_X)$, 2) $\pi \in C(T_1|_X)\Delta C(T_2|_X)$, 3) $\pi \in C(T_1|_X) \cap C(T_2|_X)$.

Case 1): Let $\pi \notin C(T_1|_X) \cup C(T_2|_X)$. Since no edge induces π in $T_1|_X$ or $T_2|_X$, we have $w(\pi) = 0$. Assume for contradiction that there exists a bipartition $\pi' \in (C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X$ for some $i \in \{1,2\}$. Since $\pi \notin C(T_1|_X) \cup C(T_2|_X)$ and $\pi'|_X = \pi$, by Corollary 2, $\pi' \notin C(T_i)$ for any $i \in \{1,2\}$. This contradicts with the fact that $\pi' \in C(T_i)$ for some $i \in \{1,2\}$. Therefore, the assumption that there exists such a bipartition π' is wrong and $\sum_{i=1,2} |(C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X| = 0 \leq w(\pi)$.

Case 2): Let $\pi \in C(T_1|_X)\Delta C(T_2|_X)$. Assume without loss of generality that $\pi \in C(T_1|_X)\backslash C(T_2|_X)$. Then, we have $w(\pi) = w(e_1)$. Let $\pi' \in (C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X$ for some $i \in \{1,2\}$. Since $\pi'|_X = \pi$ and $\pi \notin C(T_2|_X)$, by Corollary 2, we have $\pi' \notin C(T_2)$. Since $\pi' \in C(T_i)$ for some $i \in \{1,2\}$, it must be that $\pi' \in C(T_1)$. By Lemma 1, the edge which induces π' in T_1 is an edge on $P_1(e_1)$. Since there are $w(e_1)$ edges on $P_1(e_1)$, there are at most $w(e_1)$ distinct such bipartitions π's, and thus the statement is proved.

Case 3): Let $\pi \in C(T_1|_X) \cap C(T_2|_X)$. Then we have $w(\pi) = w(e_1) + w(e_2)$. Fix any $\pi' \in (C(T'|_{S_1})\backslash C(T|_{S_1})) \cap C(T_1) \cap \Pi_X$. Since $\pi' \in C(T_1)$ and $\pi'|_X = \pi \in C(T_1|_X)$, by Lemma 1, the edge e' that induces π' is an edge on $P_1(e_1)$. Recall that $w(e_1) = |P_1(e_1)|$, then we have $|(C(T'|_{S_1})\backslash C(T|_{S_1})) \cap C(T_1) \cap \Pi_X| \leq$

$|P_1(e_1)| = w(e_1)$. Similarly, $|(C(T'|_{S_2})\backslash C(T|_{S_2})) \cap C(T_2) \cap \Pi_X| \leq |P_2(e_2)| = w(e_2)$. Therefore, $\sum_{i=1,2} |(C(T'|_{S_i})\backslash C(T|_{S_i})) \cap C(T_i) \cap \Pi_X| \leq w(\pi)$. □

Lemma 5. *For any compatible set F of bipartitions of X, let T be a tree of leaf set S such that $C(T|_X) = F$. Then $p_X(T) \leq \sum_{\pi \in F} w(\pi)$.*

Proof. Fix an arbitrary ordering of bipartitions in F and let them be $\pi_1, \pi_2, \ldots, \pi_k$, where $k = |F|$. Let $F_i = \{\pi_1, \ldots, \pi_i\}$ for any $i \in \{0, 1, \ldots, k\}$. In particular, $F_0 = \emptyset$ and $F_k = F$. Let T^i be obtained by contracting any edge e in T such that $\pi_e \in \Pi_X$ and $\pi_e|_X \notin F_i$. Then $C(T^i|_X) = F_i$. In particular, we know $C(T^0|_X) = \emptyset$. By construction, T^i is a refinement of T^{i-1} for any $i \in \{1, 2, \ldots, k\}$ such that for any $\pi' \in C(T^i)\backslash C(T^{i-1})$, $\pi'|_X = \pi_i$. Then by Lemma 4, $p_X(T^i) - p_X(T^{i-1}) \leq w(\pi_i)$. Therefore,

$$p_X(T) - p_X(T^0) = \sum_{i=1}^{k} p_X(T^i) - p_X(T^{i-1}) \leq \sum_{i \in F} w(\pi_i).$$

We also know that $p_X(T^0) = 0$ (expand on this) and thus $p_X(T) \leq \sum_{i \in I} w(\pi_i)$ as desired. □

Claim 3. *Let $\tilde{T}$ be the tree constructed in Algorithm 1, then $p_X(\tilde{T}) = \sum_{\pi \in \mathrm{Triv}} w(\pi)$.*

Lemma 6. *Let T be a tree from Algorithm 1 before a refinement step. Let $\pi = A|B \in \mathrm{NonTriv} \cap (I \cup (C(T_1|_X) \cap C(T_2|_X)))$. Let T' be a refinement of T obtained from running Algorithm 3 on T and π, with the auxiliary data structures H, sv, and $\mathcal{T}$. Then, $p_X(T') - p_X(T) = w(\pi)$.*

Proof. The algorithm makes sure that $\pi \notin C(T|_X)$ and all bipartitions in $C(T|_X)$ are compatible with π □

Let G be the incompatibility graph defined in Algorithm 1 and I be the maximum weight independent set in G with weight function w. Let $G' = (V_1' \cup V_2', E')$ be another incompatibility graph such that $V_1' = C(T_1|_X)$ and $V_2' = C(T_2|_X)$, and $E' = \{(\pi, \pi') \mid \pi \in V_1', \pi' \in V_2',\ \pi \text{ is not compatible with } \pi'\}$. Let $I' := I \cup (C(T_1|_X) \cap C(T_2|_X))$.

Claim 4. *I' is a maximum weight independent set in G' with weight function w.*

Proof. □

Claim 5. *Let T^* be the tree defined in Algorithm 1, $p_X(T^*) \geq p_X(T)$ for any tree T of leafset S.*

Proof. then $p_X(T^*) = \sum_{\pi \in I} w(\pi) + \sum_{\pi \in (C(T_1|_X) \cap C(T_2|_X))} w(\pi) = \sum_{\pi \in I'} w(\pi)$. □

Theorem 2. *Algorithm 1 correctly solves* Max-Bisup-Supertree-2 *in $O(n^3)$ (Checking this) time.*

Proof. From Lemma 3 and Claim 5 and Lemma 2, we know that T^* defined in Algorithm 1 satisfy that $p_X(T^*) \geq p_X(T)$ and $p_Y(T^*) \geq p_Y(T)$ for any tree with leaf set S. Then by Claim 1, Algorithm 1 correctly solves Max-Bisup-Supertree-2. Next we analyze the running time of Algorithm 1. $\square$

2.4 Hardness for Max-Bisup-Supertree-3

In this subsection, we show that Max-Bisup-Supertree-N is NP-hard even when $N = 3$. We reduce the maximum weight independent set problem on tripartite graphs to Max-Bisup-Supertree-3. We first reproduce the theorem that proves the maximum weight independent set problem on tripartite graphs to be NP-hard.

Theorem 3. Max-Bisup-Supertree-3 *is NP-hard.*

A Proofs from Section 2

Proof of Lemma 1

Proof. Let T_R be the minimal subtree of T that spans R. It follows that the leaf set of T_R is R and $T|_R$ is obtained from T_R by suppressing all degree-two nodes. Let $\pi' = A'|B'$. By definition of e inducing $\pi = A|B$, the vertices of A are all disconnected from vertices of B in $T - e$. If $R \cap A \neq \emptyset$ and $R \cap B \neq \emptyset$, then e is necessary to connect $R \cap A$ with $R \cap B$, and thus e must be in any tree spanning R and in particular $e \in E(T_R)$. Since T_R is a subgraph of T, the two components in $T_R - e$ are subgraphs of the two components in $T - e$. Thus, the leaves of the two components in $T_R - e$ are exactly $R \cap A$ and $R \cap B$. We also know that suppressing degree-two nodes does not change the connectivity between any leaves so the leaves of the two components in $T_R - P(e')$ (with vertices on the path also deleted) are the same as the leaves of the two components in $T|_R - e'$, which are A' and B'. If $e \in P(e')$, since all internal nodes of $P(e')$ have degree two with both incident edges on $P(e')$, there is no leaf which exists in any of the two components in $T_R - e$ but does not exists in the corresponding component in $T_R - P(e')$. Therefore, $\pi|_R = R \cap A | R \cap B = A'|B' = \pi'$. If $e \notin P(e')$, then since $e \in E(T_R)$, there must exists $e'' \in E(T|_R)$ such that $e'' \neq e'$ and $e \in P(e'')$. By the arguement above, $\pi|_R = \pi''$ where π'' is the bipartition induced by e'' in $T|_R$. Since $e'' \neq e'$, we know $\pi' \neq \pi''$ and thus $\pi|_R \neq \pi'$. This concludes our proof that $\pi|_R = \pi'$ if and only if $e \in P(e')$. $\square$

References

[1] Tandy Warnow. *Computational phylogenetics: an introduction to designing methods for phylogeny estimation.* Cambridge University Press, 2017.