Finding Group Steiner Trees in Graphs with both Vertex and Edge Weights: Some Supplemental Materials

Dear reviewers,

We are extremely grateful for your careful reading and constructive comments. Below are our responses to the comments.

1 THE TRANSFORMATION

THEOREM 1.1. *Let $G(V, E, w, c)$ be a connected undirected graph, and Γ be a set of vertex groups. Let $G_t(V_t, E_t, w_t, c_t)$ be a connected undirected graph, and $T_t \subseteq V_t$ be a set of compulsory vertices. Based on G and Γ, we construct G_t and T_t in the following way:*

(1) Initialize $V_t = V$, $E_t = E$, $T_t = \emptyset$, $w_t = (1-\lambda)w$, and $c_t = \lambda c$.

(2) For each vertex group $g \in \Gamma$, (i) add a dummy vertex v_g into T_t and V_t, such that $w_t(v_g) = 0$, and (ii) add dummy edges (v_g, j) for all $j \in g$ into E_t, such that $c_t(v_g, j) = M$, and M is a constant satisfying

$$M > (1-\lambda)\sum_{v \in V} w(v) + \lambda \sum_{e \in E_{MST}} c(e), \tag{1}$$

and E_{MST} is the set of edges in a Minimum Spanning Tree of G.

Let Θ_{G_t} be an optimal solution to the Steiner tree problem in G_t, and $\Theta^{non}_{G_t}$ be the non-dummy part of Θ_{G_t}. Then, there is an optimal solution to the group Steiner tree problem in G, namely, Θ_G, that has the same sets of vertices and edges with $\Theta^{non}_{G_t}$.

PROOF. Since dummy vertices only connect non-dummy vertices, there are at least $|\Gamma|$ dummy edges in Θ_{G_t}. If $c_\lambda(\Theta_G) < c(\Theta^{non}_{G_t})$, then there is a feasible solution to the Steiner tree problem in G_t: Θ'_{G_t} such that

$$c(\Theta'_{G_t}) = c_\lambda(\Theta_G) + M|\Gamma| < c(\Theta_{G_t}), \tag{2}$$

which is not possible. Thus, we have $c_\lambda(\Theta_G) \geq c(\Theta^{non}_{G_t})$. Let Θ''_{G_t} be a tree in G_t such that (i) every dummy vertex v_g is a leaf of Θ''_{G_t}; and (ii) the non-dummy part of Θ''_{G_t}, namely, $\Theta^{non''}_{G_t}$, is in a Minimum Spanning Tree of G. Suppose that there is a dummy vertex v_g in Θ_{G_t} that is not a leaf. Since $c(\Theta^{non''}_{G_t}) < M$, we have

$$c(\Theta_{G_t}) \geq c(\Theta^{non}_{G_t}) + M(|\Gamma|+1) > c(\Theta''_{G_t}) = c(\Theta^{non''}_{G_t}) + M|\Gamma|, \tag{3}$$

which is not possible. Thus, every dummy compulsory vertex v_g is a leaf of Θ_{G_t}. As a result, $\Theta^{non}_{G_t}$ is connected and shares the same sets of vertices and edges with a feasible solution to the group Steiner tree problem in G, which means that $c_\lambda(\Theta_G) \leq c(\Theta^{non}_{G_t})$. Therefore, $c_\lambda(\Theta_G) = c(\Theta^{non}_{G_t})$. Hence, this theorem holds. □

2 THE APPROXIMATION GUARANTEE OF LANCET

LANCET can be regarded as the vertex- and edge-weighted version of the algorithm in [4], which achieves an approximation guarantee of $2(1 - 1/|T_t|)$ for solving the vertex-unweighted Steiner tree problem. This approximation guarantee relies on the following deduction (*i.e.,* Lemma 1 in [4]): since a pre-order traversal of a tree traverses every edge in this tree exactly twice, in a graph with only edge weights, if we perform a pre-order traversal of an optimal solution tree and sum up every weight that we encounter (including duplicates), then the result is exactly twice the weight of an optimal solution tree. However, in a graph with both vertex and edge weights, summing up the weights that we encounter during this traversal does not always result in twice the weight of an optimal solution tree, since (i) an optimal solution tree may contain non-compulsory vertices with positive weights; and (ii) a pre-order traversal of an optimal solution tree may visit such a vertex more than twice (specifically, the number of times that a pre-order traversal of an optimal solution tree visits such a vertex equals the degree of this vertex in this optimal solution tree). Thus, the above approximation guarantee of $2(1 - 1/|T_t|)$ does not hold for LANCET. In what follows, we establish the approximation guarantee of LANCET.

THEOREM 2.1. LANCET *has a sharp approximation guarantee of $|T_t| - 1$ for solving the vertex- and edge-weighted Steiner tree problem.*

PROOF. Since $|V_{min} \cap V_2| = 1$ (Line 6 in LANCET), LANCET merges $|T_t| - 1$ LWPs to connect all compulsory vertices together. Suppose that the highest-weight one of these LWPs is LWP', and Θ_{opt} is an optimal solution. Since $c(LWP')$ is smaller than or equal to the weight of the LWP between a pair of compulsory vertices, we have

$$c(\Theta_{opt}) \geq c(LWP'). \tag{4}$$

Since there are $|T_t| - 1$ LWPs that have been merged, we have

$$(|T_t| - 1)c(\Theta_{opt}) \geq (|T_t| - 1)c(LWP') \geq c(\Theta). \tag{5}$$

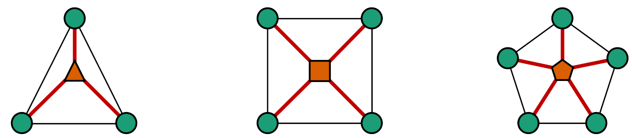

Figure 1: Touching the approximation guarantee of $|T_t| - 1$.

Therefore, LANCET has an approximation guarantee of $|T_t| - 1$. We further show that $|T_t| - 1$ is the sharp approximation guarantee of LANCET. Consider a regular polygon composed of $|T_t|$ compulsory vertices, and a non-compulsory vertex that is in the middle of this polygon and connects $|T_t|$ compulsory vertices. The weight of each edge between compulsory vertices is x, the weight of each edge between a compulsory vertex and the middle non-compulsory vertex is 0, the weight of each compulsory vertex is 0, and the weight of the middle non-compulsory vertex is z. Suppose that $x = z - \delta$, where δ is a tiny positive value; and $z < (|T_t| - 1)x$. Since $x < z$, Θ contains $|T_t| - 1$ edges between compulsory vertices, and $c(\Theta) = (|T_t| - 1)x$. Since $z < (|T_t| - 1)x$, Θ_{opt} contains all the edges between compulsory vertices and the middle non-compulsory vertex, and $c(\Theta_{opt}) = z$. We have

$$\lim_{\delta \to 0} \frac{c(\Theta)}{c(\Theta_{opt})} = \frac{(|T_t| - 1)(z - \delta)}{z} = |T_t| - 1. \tag{6}$$

Hence, $|T_t| - 1$ is the sharp approximation guarantee of LANCET.

□

3 THE APPROXIMATION GUARANTEE OF exIhlerA

THEOREM 3.1. exIhlerA *has a sharp approximation guarantee of* $|\Gamma| - 1$ *for solving the vertex- and edge-weighted group Steiner tree problem.*

PROOF. Suppose that $\Theta_{OPT}(V_{OPT}, E_{OPT})$ is an optimal solution. Let $\Gamma = \{g_1, \ldots, g_{|\Gamma|}\}$. There is a tuple $(v_1, \ldots, v_{|\Gamma|})$ such that $v_i \in V_{OPT} \cap g_i$ for all $i \in \{1, \ldots, |\Gamma|\}$. Without loss of generality, assume that $g_{min} = g_1$. For every $i \in \{2, \ldots, |\Gamma|\}$, there is exactly one simple path between v_1 and v_i in Θ_{OPT}, which we refer to as $P_{v_1 v_i}$. We have

$$c_\lambda(P_{v_1 v_i}) \leq c_\lambda(\Theta_{OPT}), \tag{7}$$

$$\sum_{g \in \Gamma \setminus g_1} c_\lambda(LWP_{\lambda v_1 g}) \leq \sum_{i \in \{2, \ldots, |\Gamma|\}} c_\lambda(P_{v_1 v_i}). \tag{8}$$

Thus,

$$c_\lambda(\Theta) \leq c_\lambda(G_{v_1}) \leq \sum_{g \in \Gamma \setminus g_1} c_\lambda(LWP_{\lambda v_1 g}) \leq \sum_{i \in \{2, \ldots, |\Gamma|\}} c_\lambda(P_{v_1 v_i}) \leq (|\Gamma| - 1) c_\lambda(\Theta_{OPT}). \tag{9}$$

Hence, exIhlerA has an approximation guarantee of $|\Gamma| - 1$. We note that this guarantee is sharp. To explain, consider the graph $G(V, E, w, c)$ in Figure 2, where $V = \{v_0, v_1, \ldots, v_{|\Gamma|}\}$, $E = \{(v_{|\Gamma|}, v_0), (v_{|\Gamma|}, v_1), \ldots, (v_{|\Gamma|}, v_{|\Gamma|-1})\}$, $w(i) = 0$ for all $i \in V$, $c(v_{|\Gamma|}, v_1) = \ldots = c(v_{|\Gamma|}, v_{|\Gamma|-1}) = 1$, and $c(v_{|\Gamma|}, v_0) = 1 + \delta$, where δ is a tiny positive value. In addition, $\Gamma = \{v_0, v_1\} \cup \ldots \cup \{v_0, v_{|\Gamma|-1}\} \cup \{v_{|\Gamma|}\}$. Let $\lambda = 1$. Since $g_{min} = \{v_{|\Gamma|}\}$, exIhlerA produces the solution $\Theta = \{(v_{|\Gamma|}, v_1), \ldots, (v_{|\Gamma|}, v_{|\Gamma|-1})\}$, and $c_\lambda(\Theta) = |\Gamma| - 1$. When $|\Gamma| = 2$, Θ is the optimal solution, *i.e.*, the approximation ratio is $|\Gamma| - 1 = 1$. When $|\Gamma| > 2$, we have $\Theta_{OPT} = \{(v_{|\Gamma|}, v_0)\}$, and

$$\lim_{\delta \to 0} \frac{c_\lambda(\Theta_B)}{c_\lambda(\Theta_{OPT})} = \frac{|\Gamma| - 1}{1 + \delta} = |\Gamma| - 1. \tag{10}$$

Hence, the best possible approximation guarantee of exIhlerA is $|\Gamma| - 1$. □

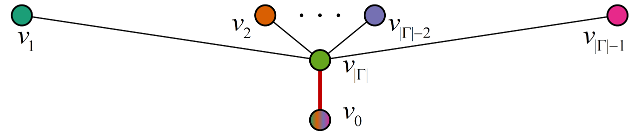

Figure 2: Touching the approximation guarantee of $|\Gamma| - 1$.

4 THE APPROXIMATION GUARANTEE OF FastAPP

THEOREM 4.1. FastAPP *has a sharp approximation guarantee of* $|\Gamma| - 1$ *for solving the vertex- and edge-weighted group Steiner tree problem.*

PROOF. Let $\Theta_{OPT}(V_{OPT}, E_{OPT})$ be an optimal solution, and $\Gamma = \{g_1, \ldots, g_{|\Gamma|}\}$. There is a tuple $(v_1, \ldots, v_{|\Gamma|})$ such that $v_i \in V_{OPT} \cap g_i$ for all $i \in \{1, \ldots, |\Gamma|\}$. Without loss of generality, suppose that $g_{min} = g_1$. Let $g_x \in \Gamma \setminus g_1$ be such a vertex group that

$$c_\lambda(LWP_{\lambda v_1 g_x}) = \max\{c_\lambda(LWP_{\lambda v_1 g}) \mid \forall g \in \Gamma \setminus g_1\}. \tag{11}$$

Since $LWP_{\lambda v_1 g_x}$ links fewer groups to v_1 than Θ_{OPT}, we have

$$c_\lambda(LWP_{\lambda v_1 g_x}) \leq c_\lambda(\Theta_{OPT}). \tag{12}$$

Lines 5-8 in FastAPP guarantee that

$$\max\{c_\lambda(LWP_{\lambda i_{min} g}) \mid \forall g \in \Gamma \setminus g_1\} \leq c_\lambda(LWP_{\lambda v_1 g_x}). \tag{13}$$

By Lines 10-11 in FastAPP, we have

$$c_\lambda(\Theta) \leq (|\Gamma| - 1) \max\{c_\lambda(LWP_{\lambda i_{min} g}) \mid \forall g \in \Gamma \setminus g_1\}. \tag{14}$$

Thus,

$$c_\lambda(\Theta) \leq (|\Gamma| - 1) c_\lambda(LWP_{\lambda v_1 g_x}) \leq (|\Gamma| - 1) c_\lambda(\Theta_{OPT}). \tag{15}$$

Hence, FastAPP has an approximation guarantee of $|\Gamma| - 1$. The sharpness of this guarantee can be seen from the example in Section 3, *i.e.,* Figure 2. Hence, this theorem holds. □

5 THE APPROXIMATION GUARANTEE OF ImprovAPP

THEOREM 5.1. ImprovAPP *has a sharp approximation guarantee of* $|\Gamma| - 1$ *for solving the vertex- and edge-weighted group Steiner tree problem.*

PROOF. Let $\Theta_{OPT}(V_{OPT}, E_{OPT})$ be an optimal solution. Let $\Gamma = \{g_1, \ldots, g_{|\Gamma|}\}$. There is a tuple $(v_1, \ldots, v_{|\Gamma|})$ such that $v_i \in V_{OPT} \cap g_i$ for all $i \in \{1, \ldots, |\Gamma|\}$. Without loss of generality, suppose that $g_{min} = g_1$. When ImprovAPP processes v_1 in the for loop (Lines 5-14), it concatenates $|\Gamma| - 1$ lowest weight paths that link $\{g_2, \ldots, g_{|\Gamma|}\}$, respectively, to build Θ_{v_1}. Let $LWP_{\lambda v_x g_y}$ be one of these paths that has the largest regulated weight, and links $g_y \in \{g_2, \ldots, g_{|\Gamma|}\}$. Thus,

$$c_\lambda(\Theta_{v_1}) \leq (|\Gamma| - 1) c_\lambda(LWP_{\lambda v_x g_y}). \tag{16}$$

Let $LWP_{\lambda v_1 g_y}$ be the lowest weight path between v_1 and g_y. Since $LWP_{\lambda v_1 g_y}$ has been pushed into Q initially (Line 6) and has (possibly) been updated to $LWP_{\lambda v_x g_y}$ (Line 12), we have

$$c_\lambda(LWP_{\lambda v_x g_y}) \leq c_\lambda(LWP_{\lambda v_1 g_y}). \tag{17}$$

Since $LWP_{\lambda v_1 g_y}$ links fewer groups to v_1 than Θ_{OPT}, we have

$$c_\lambda(LWP_{\lambda v_1 g_y}) \leq c_\lambda(\Theta_{OPT}). \tag{18}$$

Thus,

$$c_\lambda(\Theta) \leq c_\lambda(\Theta_{v_1}) \leq (|\Gamma| - 1) c_\lambda(LWP_{\lambda v_x g_y}) \leq (|\Gamma| - 1) c_\lambda(LWP_{\lambda v_1 g_y}) \leq (|\Gamma| - 1) c_\lambda(\Theta_{OPT}). \tag{19}$$

Therefore, ImprovAPP has an approximation guarantee of $|\Gamma| - 1$. The sharpness of this guarantee can also be seen from the example in Section 3, *i.e.,* Figure 2. Thus, this theorem holds. □

6 THE APPROXIMATION GUARANTEE OF PartialOPT

THEOREM 6.1. PartialOPT *has a sharp approximation guarantee of* $|\Gamma| - h + 1$ *for solving the vertex- and edge-weighted group Steiner tree problem.*

PROOF. Suppose that $\Theta_{OPT}(V_{OPT}, E_{OPT})$ is an optimal solution, and $\Gamma = \{g_1, \ldots, g_{|\Gamma|}\}$. There is a tuple $(v_1, \ldots, v_{|\Gamma|})$ such that $v_i \in V_{OPT} \cap g_i$ for all $i \in \{1, \ldots, |\Gamma|\}$. Without loss of generality, let $g_{min} = g_1$. For $v_1 \in g_{min}$, $\Theta^h_{v_1}$ is optimal for $\Gamma_1 = \{\{v_1\}, g_2, \ldots, g_h\}$. Since $\Theta^h_{v_1}$ connects fewer groups, we have

$$c_\lambda(\Theta^h_{v_1}) \leq c_\lambda(\Theta_{OPT}). \tag{20}$$

Subsequently, we implement exIhlerA to produce $\Theta^{|\Gamma|}_{v_1}$ for $\Gamma_2 = \{\{v_1\}, g_{h+1}, \ldots, g_{|\Gamma|}\}$. Suppose that $\Theta^{|\Gamma|}_{OPT}$ is an optimal solution for Γ_2. The proof of Theorem 3.1 shows that

$$c_\lambda(\Theta^{|\Gamma|}_{v_1}) \leq (|\Gamma| - h) c_\lambda(\Theta^{|\Gamma|}_{OPT}). \tag{21}$$

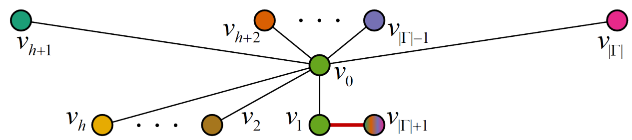

Figure 3: Touching the approximation guarantee of $|\Gamma| - h + 1$.

Since $\Theta_{OPT}^{|\Gamma|}$ connects fewer groups, we have

$$c_\lambda(\Theta_{OPT}^{|\Gamma|}) \leq c_\lambda(\Theta_{OPT}). \tag{22}$$

Thus,

$$c_\lambda(\Theta) \leq c_\lambda(G_{v_1}) = c_\lambda(\Theta_{v_1}^h \cup \Theta_{v_1}^{|\Gamma|}) \leq c_\lambda(\Theta_{v_1}^h) + c_\lambda(\Theta_{v_1}^{|\Gamma|}) \leq (|\Gamma| - h + 1)c_\lambda(\Theta_{OPT}). \tag{23}$$

Therefore, PartialOPT has an approximation guarantee of $|\Gamma| - h + 1$. We show that this guarantee is sharp. Consider the graph $G(V, E, w, c)$ in Figure 3, where $V = \{v_0, v_1, \ldots, v_{|\Gamma|+1}\}$, $E = \{(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_{|\Gamma|}), (v_1, v_{|\Gamma|+1})\}$, $w(i) = 0$ for every $i \in \{v_0, \ldots, v_{h-1}\}$, $w(i) = 1$ for every $i \in \{v_h, \ldots, v_{|\Gamma|+1}\}$, $c(v_0, v_1) = \delta_1$, $c(v_1, v_{|\Gamma|+1}) = \delta_2$, where δ_1 and δ_2 are two tiny positive values, and $\delta_1 < \delta_2$, and all the other edge weights are zero. In addition, $\Gamma = \{g_1, \ldots, g_{|\Gamma|}\} = \{v_0, v_1\} \cup \{v_2, v_{|\Gamma|+1}\} \cup \ldots \cup \{v_{|\Gamma|}, v_{|\Gamma|+1}\}$. Let $\lambda = 0.5$, *i.e.,* vertex and edge weights are regulated equally. PartialOPT enumerates two vertices in g_{min}, *i.e.,* v_0 and v_1. For v_0, PartialOPT produces $\Theta_{v_0}^h = \{(v_0, v_2), \ldots, (v_0, v_h)\}$, and $\Theta_{v_0}^{|\Gamma|} = \{(v_0, v_{h+1}), \ldots, (v_0, v_{|\Gamma|})\}$. Thus, $\Theta_{v_0} = \{(v_0, v_2), \ldots, (v_0, v_{|\Gamma|})\}$. Similarly, for v_1, since $\delta_1 < \delta_2$, PartialOPT produces $\Theta_{v_1} = \{(v_0, v_1), \ldots, (v_0, v_{|\Gamma|})\}$. We have $\Theta = \Theta_{v_0}$. When $|\Gamma| = h$, Θ is the optimal solution, *i.e.,* the approximation ratio is $|\Gamma| - h + 1 = 1$. When $|\Gamma| > h$, we have $\Theta_{OPT} = \{(v_1, v_{|\Gamma|+1})\}$, and

$$\lim_{\delta_2 \to 0} \frac{c_\lambda(\Theta)}{c_\lambda(\Theta_{OPT})} = \frac{|\Gamma| - h + 1}{1 + \delta_2} = |\Gamma| - h + 1. \tag{24}$$

Hence, $|\Gamma| - h + 1$ is the best possible approximation guarantee of PartialOPT. This theorem holds. □

7 THE RECENT WORK ON IMPROVING DPBF

The recent work [3] improves the dynamic programming algorithm DPBF [1] for finding optimal vertex-unweighted group Steiner trees. The main idea of this improvement is to incorporate pruning techniques into the process of DPBF. There are three algorithms in [3]: Basic, PrunedDP and PrunedDP++. Here, we investigate these three algorithms for finding optimal vertex- and edge-weighted group Steiner trees. First, in Section 7.1, we show that Basic can be extended to find optimal vertex- and edge-weighted group Steiner trees, but is slower than DPBF. Then, in Section 7.2, we show that PrunedDP and PrunedDP++ depend on pruning techniques that do not hold in graph with vertex weights.

7.1 Basic can find optimal vertex- and edge-weighted group Steiner trees

Let $T(v, \Gamma)$ be the minimum-weight group Steiner tree that roots at vertex v and covers all vertex groups in Γ. DPBF is mainly based on the following observation: the minimum-weight group Steiner tree that roots at vertex v and covers a contain set of vertex groups can be constructed by merging the minimum-weight group Steiner tree that roots at vertex v and covers part of these vertex groups. Based on this observation, DPBF dynamically enumerates $T(v, \Gamma')$ for all $v \in V$ and $\Gamma' \subseteq \Gamma$, in an increasing order of the weight of $T(v, \Gamma')$, until the minimum-weight group Steiner tree that covers all vertex groups in Γ is found. Basic enhances DPBF by pruning unpromising $T(v, \Gamma')$ in the following way.

When enumerating $T(v, \Gamma')$, Basic constructs a feasible solution tree by (i) merging shortest paths between v and each vertex group in $\Gamma \setminus \Gamma'$ into $T(v, \Gamma')$; and (ii) finding a Minimum Spanning Tree (MST) of the merged graph as a feasible solution tree (this idea of constructing a feasible solution tree is similar to IhlerA [2]). The weight of this feasible solution tree can be seen as an upper bound of the weight of an optimal solution tree. With this upper bound, Basic prunes $T(v, \Gamma'')$ if the weight of $T(v, \Gamma'')$ is larger than the weight of this upper bound, since an optimal solution tree cannot be constructed by merging $T(v, \Gamma'')$.

We can extend Basic to find optimal vertex- and edge-weighted group Steiner trees by replacing the above shortest paths with lowest-weight paths, *i.e.,* the paths that minimize the total vertex and edge weights. We compare DPBF with Basic for finding optimal vertex- and edge-weighted group Steiner trees in Figure **. It can be seen that Basic can find optimal vertex- and edge-weighted group Steiner trees, but is slower than DPBF. The reason is that, in the worst case, Basic constructs a feasible solution tree for each $T(v, \Gamma')$ by merging lowest-weight paths and finding an MST. As a result, the worst-case time complexity of Basic is

$$O\Big(3^{|\Gamma|}|V| + 2^{|\Gamma|}|V| \cdot (|\Gamma||V| + |E| + |V| \log |V|)\Big),$$

where $O(3^{|\Gamma|}|V|)$ corresponds to the time complexity of the merging process in DPBF and Basic (details in [1]), $O(2^{|\Gamma|}|V| \cdot |\Gamma||V|)$ corresponds to the time complexity of merging $O(|\Gamma|)$ lowest-weight paths into every $T(v, \Gamma')$ for constructing feasible solution trees (details in [3]),

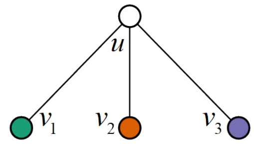

Figure 4: A toy instance for showing that Theorem 2 in [3] does not hold in graphs with vertex weights.

and $O(2^{|\Gamma|}|V| \cdot (|E| + |V| \log |V|))$ corresponds to finding an MST when enumerating every $T(v, \Gamma')$ for constructing feasible solution trees (details in [3]). In comparison, the worst-case time complexity of DPBF is (details in [1])

$$O\Big(3^{|\Gamma|}|V| + 2^{|\Gamma|} \cdot (|\Gamma||V| + |E| + |V| \log |V|)\Big).$$

7.2 Pruning techniques that do not allow vertex weights in PrunedDP and PrunedDP++

Here, we show that PrunedDP and PrunedDP++ depend on pruning techniques that do not hold in graphs with vertex weights.

Theorem 2 in [3] does not allow vertex weights. First, note that Theorem 2 in [3] is the core pruning technique in PrunedDP, and is also an important pruning technique in PrunedDP++. This theorem does not hold in graphs with vertex weights. To explain, we first briefly describe the dynamic programming process of DPBF through a toy example in Figure 4. Understanding this process is necessary for understanding the reason why Theorem 2 in [3] does not hold in graphs with vertex weights.

In Figure 4, there are three vertex groups $g_1 = \{v_1\}$, $g_2 = \{v_2\}$ and $g_3 = \{v_3\}$. The weight of u is 1, and each of the other vertex and edge weights is δ, and δ is a tiny positive value. The optimal solution tree is the whole graph, and the weight of the optimal solution tree is $1 + 6\delta$. To find this tree, DPBF first initializes $T(v_1, \{g_1\})$ as the single vertex v_1; $T(v_2, \{g_2\})$ as the single vertex v_2; and $T(v_3, \{g_3\})$ as the single vertex v_3. Then, DPBF grows $T(v_1, \{g_1\})$, $T(v_2, \{g_2\})$ and $T(v_3, \{g_3\})$ to vertex u, and produces $T(u, \{g_1\})$ as the edge (u, v_1); $T(u, \{g_2\})$ as the edge (u, v_2); and $T(u, \{g_3\})$ as the edge (u, v_3). Subsequently, it merges $T(u, \{g_1\})$ and $T(u, \{g_2\})$ as $T(u, \{g_1, g_2\}) = \{(u, v_1), (u, v_2)\}$. At last, it merges $T(u, \{g_3\})$ and $T(u, \{g_1, g_2\})$, and produces the optimal solution tree.

Theorem 2 in [3] proves that: if all vertex weights are zero, then in DPBF, we can merge two subtrees $T(u, \Gamma')$ and $T(u, \Gamma'')$ for $\Gamma'' \subset \Gamma \setminus \Gamma'$ only when the total weight of these two subtrees is no larger than $\frac{2}{3}$ of the weight of an optimal solution tree. For example, if all vertex weights are zero in the above instance, then the weight of the optimal solution tree is 3δ. When we merge $T(u, \{g_1\})$ and $T(u, \{g_2\})$ as $T(u, \{g_1, g_2\})$ in the above process, the total weight of $T(u, \{g_1\})$ and $T(u, \{g_2\})$ is 2δ, which is no larger than $\frac{2}{3}$ of the weight of an optimal solution tree. By Theorem 2 in [3], merging these two subtrees may help produce the optimal solution tree. If the total weight of $T(u, \{g_1\})$ and $T(u, \{g_2\})$ is larger than $\frac{2}{3}$ of the weight of an optimal solution tree, then merging these two subtrees does not help produce the optimal solution tree, and thus this merge can be avoided. However, this is true only when all vertex weights are zero. For example, if we consider the vertex weights in the above instance, then the total weight of $T(u, \{g_1\})$ and $T(u, \{g_2\})$ is $1 + 4\delta$, which is larger than $\frac{2}{3}$ of the weight of an optimal solution tree: $1 + 6\delta$ (in worst scenarios where $\delta = 0$, the total weight of $T(u, \{g_1\})$ and $T(u, \{g_2\})$ equals the weight of an optimal solution tree). As a result, if we use Theorem 2 in [3] in the above instance with vertex weights, then the optimal solution tree will never be found. That is to say, Theorem 2 in [3] does not hold in graphs with vertex weights.

We point out the specific place in the proof of Theorem 2 in [3] that does not hold in graphs with vertex weights as follows. In the beginning of the proof of Theorem 2 in [3], an optimal solution is assumed to be a tree rooted at vertex u with k subtrees, $T_1, \ldots, T_k$, that root at v_i respectively and the weight of each of these subtrees is smaller than half of the weight of an optimal solution tree (*e.g.,* in Figure 4, T_i is the single vertex v_i). Let $\overline{T_i}$ be the the edge-grown subtree that is grown by T_i with an edge (v_i, u) (*e.g.,* in Figure 4, $\overline{T_i}$ is the edge (v_i, u)). The proof of Theorem 2 in [3] then claims that: there are three different cases: (1) the weight of each $\overline{T_i}$ is smaller than half of the weight of an optimal solution tree; (2) there is only one edge-grown subtree $\overline{T_i}$ that has a weight no smaller than half of the weight of an optimal solution tree; and (3) there are two edge-grown subtrees and the weight of each one is half of the weight of an optimal solution tree. This claim is not true in graphs with vertex weights, where there is a fourth case: there are more than two edge-grown subtrees and the weight of each one is large than half of the weight of an optimal solution tree (*e.g.,* in Figure 4, due to the existence of vertex weights, the weight of each $\overline{T_i}$ is $1 + 2\delta$, which is larger than half of the weight of an optimal solution tree).

Since Theorem 2 in [3] is the core pruning technique in PrunedDP and does not hold in graphs with vertex weights, we do not implement PrunedDP in our paper.

Lemma 2 in [3] does not allow vertex weights. Theorem 2 in [3] is also an important pruning technique in PrunedDP++. Another important pruning technique in PrunedDP++ is the tour-based lower bounds construction method in [3]. Lemma 2 in [3] is a key of this method. In the following, we show that this lemma does not hold in graphs with vertex weights.

First, we briefly introduce the label-enhanced graph in [3], which is constructed by adding dummy vertices and edges into the graph as follows. For each group $g_i \in \Gamma$, we add a dummy vertex $\widetilde{v_i}$, and also add a dummy edge $(\widetilde{v_i}, u)$ with zero weight for every $u \in g_i$. For example, in Figure 5, where the graph contains two vertices v and u, and one edge (v, u), and there are three vertex groups $g_1 = g_2 = g_3 = \{u\}$. We add three dummy vertices $\widetilde{v_1}, \ldots, \widetilde{v_2}$ and three dummy edges $(\widetilde{v_1}, u), \ldots, (\widetilde{v_3}, u)$ for creating the label-enhanced graph.

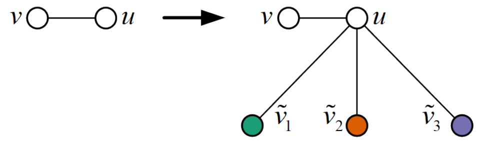

Figure 5: A toy instance for showing that Lemma 2 in [3] does not hold in graphs with vertex weights.

Then, [3] uses $W(\widetilde{v_i}, \widetilde{v_j}, \Gamma')$ to refer to the weight of the minimum weight route that starts from $\widetilde{v_i}$, ends at $\widetilde{v_j}$, and passes through all dummy vertices that corresponds to vertex groups in Γ'. Moreover, [3] uses $d(v, \widetilde{v_i})$ to refer to the weight of the minimum-weight path between non-dummy vertex v and dummy vertex $\widetilde{v_i}$. Lemma 2 in [3] is that: for any pair of vertex v and a subset of vertex groups $\Gamma' \subseteq \Gamma$, the weight of $T(v, \Gamma')$ is larger than or equal to $\frac{\min_{g_i, g_j \in \Gamma'} \{d(v,\widetilde{v_i}) + W(\widetilde{v_i}, \widetilde{v_j}, \Gamma') + d(\widetilde{v_j}, v)\}}{2}$. This is true when all vertex weights are zero. For example, in Figure 5, let the weight of edge (v, u) be δ, which is a tiny positive value, and all vertex weights be zero, and $\Gamma' = \{g_1, g_2\}$. Then, $d(v, \widetilde{v_1}) = d(\widetilde{v_2}, v) = \delta$, and $W(\widetilde{v_1}, \widetilde{v_2}, \Gamma') = 0$. As a result, $\frac{\min_{g_i, g_j \in \Gamma'} \{d(v,\widetilde{v_i}) + W(\widetilde{v_i}, \widetilde{v_j}, \Gamma') + d(\widetilde{v_j}, v)\}}{2} = \delta$. On the other hand, the weight of $T(v, \Gamma')$ is δ. Thus, Lemma 2 in [3] holds. In [3], Lemma 2 is proven by doubling every edge in $T(v, \Gamma')$ and obtaining an Euler tour that starts from v and also ends at v, and then employing the fact that the total edge weight (including duplicates) we encounter in this Euler tour is twice the total edge weight in $T(v, \Gamma')$. Nevertheless, the above observation does not hold in graphs with vertex weights. For example, Suppose that, in Figure 5, the weights of v and u are 0 and 1, respectively. Still let $\Gamma' = \{g_1, g_2\}$. Then, $d(v, \widetilde{v_1}) = d(\widetilde{v_2}, v) = 1 + \delta$, and $W(\widetilde{v_1}, \widetilde{v_2}, \Gamma') = 1$. As a result, $\frac{\min_{g_i, g_j \in \Gamma'} \{d(v,\widetilde{v_i}) + W(\widetilde{v_i}, \widetilde{v_j}, \Gamma') + d(\widetilde{v_j}, v)\}}{2} = \frac{3+2\delta}{2}$. On the other hand, the weight of $T(v, \Gamma')$ is $1 + \delta$, which is smaller than $\frac{3+2\delta}{2}$. Thus, Lemma 2 in [3] does not hold any more. The reason is that the weight of u is counted multiple times, since the tour that starts from v, to $\widetilde{v_i}$, then to $\widetilde{v_j}$, and ends at v encounters u multiple times. More generally, the reason is that the total vertex and edge weight (including duplicates) we encounter in the Euler tour in the proof of Lemma 2 in [3] may be more than twice the total vertex and edge weight in $T(v, \Gamma')$, since an Euler tour of a tree may visit a vertex in this tree more than twice. Notably, as discussed in Section 2, due to a similar reason, LANCET does not have an approximation guarantee of 2 in graphs with vertex weights.

Both Theorem 2 and Lemma 2 in [3] are core pruning techniques in PrunedDP++. Since both Theorem 2 and Lemma 2 in [3] do not hold in graphs with vertex weights, we do not implement PrunedDP++ in our paper.

8 MEMORY CONSUMPTION OF ALGORITHMS IN EXPERIMENTS

9 EFFECTIVENESS OF THE REFINEMENT PROCESS IN IMPROVAPP

REFERENCES

[1] Bolin Ding, Jeffrey Xu Yu, Shan Wang, Lu Qin, Xiao Zhang, and Xuemin Lin. 2007. Finding top-k min-cost connected trees in databases. In *IEEE International Conference on Data Engineering*. IEEE, 836–845.

[2] Edmund Ihler. 1990. Bounds on the quality of approximate solutions to the group Steiner problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 109–118.

[3] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2016. Efficient and progressive group Steiner tree search. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 91–106.

[4] Hiromitsu Takahashi and Akira Matsuyama. 1980. An approximate solution for the Steiner problem in graphs. *Math. Japonica* 24, 6 (1980), 573–577.