

Discrete Integral and Discrete Derivative on Graphs and Switch Problem of Trees

M. H. Khalifeh * and Abdol-Hossein Esfahanian

Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA; esfahanian@msu.edu

* Correspondence: khalife8@msu.edu

Abstract: For a vertex and edge weighted (VEW) graph G with a vertex weight function f_G let $W_{\alpha,\beta}(G) = \sum_{\{u,v\} \subseteq V(G)} [\alpha f_G(u) \times f_G(v) + \beta (f_G(u) + f_G(v))] d_G(u,v)$ where, $\alpha, \beta \in \mathbb{R}$ and $d_G(u,v)$ denotes the distance, the minimum sum of edge weights across all the paths connecting $u, v \in V(G)$. Assume T is a VEW tree, and $e \in E(T)$ fails. If we reconnect the two components of $T - e$ with new edge $\epsilon \neq e$ such that, $W_{\alpha,\beta}(T_{\epsilon \setminus e} = T - e + \epsilon)$ is minimum, then ϵ is called a best switch (BS) of e w.r.t. $W_{\alpha,\beta}$. We define three notions: convexity, discrete derivative, and discrete integral for the VEW graphs. As an application of the notions, we solve some BS problems for positively VEW trees. For example, assume T is an n -vertex VEW tree. Then, for the inputs $e \in E(T)$ and $w, \alpha, \beta \in \mathbb{R}^+$, we return ϵ , $T_{\epsilon \setminus e}$, and $W_{\alpha,\beta}(T_{\epsilon \setminus e})$ with the worst average time of $O(\log n)$ and the best time of $O(1)$ where ϵ is a BS of e w.r.t. $W_{\alpha,\beta}$ and the weight of ϵ is w .

Keywords: discrete derivative; discrete integral; best switch; convex graph; weighted total distance; weighted tree

MSC: 05C12; 05C22; 68Q25, 05C05; 68W05

Citation: Khalifeh, M.H.; Esfahanian, A.-H. Discrete Integral and Discrete Derivative on Graphs and Switch Problem of Trees. *Mathematics* **2023**, *11*, 1678. <https://doi.org/10.3390/math11071678>

Academic Editor: Andrea Scozzari

Received: 3 February 2023

Revised: 23 March 2023

Accepted: 27 March 2023

Published: 31 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction and Notations

Unlike Euclidean space, which is equipped with coordinate systems, we cannot stay at a network node and visualize where we are and how to achieve our goal. On the other hand, analyzing a network using local information can lower the cost of tasks [1–6]. Pre-processing may help to establish a network coordinate for local tasks and reach the destination step by step.

This paper defines discrete derivative, discrete integral, and convexity notions for vertex and edge-weighted graphs, which will help with local tasks. To do that, we choose the common definition of distance for edge-weighted graphs in the literature, which can be generalized or modified to satisfy metric properties. Applying the notions above, we design and solve some tree-related problems.

Unless otherwise stated, we assume a graph is weighted (vertices and edges) and connected without loops or multiple edges. Then, by saying that a graph G is (f_G, w_G) -weighted, the vertex weight function is f_G and the edge weight function is w_G . The weight of a path $P_G(v_1, v_n) = v_1 v_2 \dots v_n$ in G is $\sum_{i=1}^{n-1} w_G(v_i v_{i+1})$. And $d_G(u, v)$ denotes the distance between $u, v \in V(G)$, that is, the weight of the minimum weight paths across all the paths connecting u and v . Note that the distance of a vertex with itself is zero, and the distance between two vertices is infinite if they are not connected with a path. By saying a graph is positively weighted, we mean both vertex and edge weights are positive. We define the following numerical invariant for a (f_G, w_G) -weighted graph G ,

$$W_{\alpha,\beta}(G) = \sum_{\{u,v\} \subseteq V(G)} [\alpha f_G(u) \times f_G(v) + \beta (f_G(u) + f_G(v))] d_G(u,v), \quad \alpha, \beta \in \mathbb{R}.$$

And for $A, B \subseteq V(G)$, and $v \in V(G)$, we define

$$f_G(A) = \sum_{x \in A} f_G(x), \quad \sigma_G(v) = \sum_{x \in V(G)} d_G(v, x), \quad h_G(v) = \sum_{x \in V(G)} f_G(x) d_G(v, x),$$

$$d_G(A, B) = \sum_{a \in A} \sum_{b \in B} d_G(a, b).$$

And for the given subgraphs H and K of G , let:

$$d_G(H, K) = \sum_{u \in V(H)} \sum_{v \in V(K)} d_G(u, v), \quad f_G(H) = \sum_{a \in V(H)} f_G(a).$$

Definition 1. For a tree T and $uv \in E(T)$, $T - \{uv\}$ has two components. We name the components T_u^v and T_v^u with $u \in V(T_u^v)$ and $v \in V(T_v^u)$. As an extension for the later notation to subtrees, if $xy \in V(T_u^v)$, $[T_u^v]_x^y$ and $[T_u^v]_y^x$ denotes the components of $T_u^v - \{xy\}$ and so on. Also, set $n_T(T_u^v) = |V(T_u^v)|$ and $n_T(T_v^u) = |V(T_v^u)|$.

By Definition 1 $V(T) = V(T_u^v) \cup V(T_v^u)$ and $n_T(T_u^v) + n_T(T_v^u) = |V(T)|$. We remind the reader that $N_G(x)$ denotes a vertex x 's neighbors.

Using the above notation for a tree T let, $N_T = \{(n_T(T_u^v), n_T(T_v^u))\}_{uv \in E(T)}$ and $F_T = \{(f_T(T_u^v), f_T(T_v^u))\}_{uv \in E(T)}$. For more detail $n_T(T_u^v)$ and $n_T(T_v^u)$ are the number of vertices in T_u^v and T_v^u , respectively. And $f_T(T_u^v)$ and $f_T(T_v^u)$ are the total weight of vertices in T_u^v and T_v^u , respectively.

For ease and based on the definition of $W_{\alpha, \beta}$, we define two more distance-based numerical graph invariants for a (f_G, w_G) -weighted graph G as follows:

$$W_{\times}(G) = \sum_{\{u, v\} \subseteq V(G)} [f_G(u) \times f_G(v)] d_G(u, v),$$

$$W_{+}(G) = \sum_{\{u, v\} \subseteq V(G)} [f_G(u) + f_G(v)] d_G(u, v).$$

Indeed, $W_{1,0}(G) = W_{\times}(G)$, $W_{0,1}(G) = W_{+}(G)$ and $W_{\alpha, \beta}(G) = \alpha W_{\times}(G) + \beta W_{+}(G)$.

Using the notations, the coordinate of T is shown and defined with

$$Q_T = \{F_T, N_T, W_{\times}(T), W_{+}(T)\}$$

Assume T is a (f_T, w_T) -weighted tree and $e \in E(T)$. If e fails (removed from the edge set), then we have $T - e$ that has two components. If we reconnect the two components of $T - e$ with an edge, ϵ , we have $T + \epsilon - e$. For ease, we will denote $T + \epsilon - e$ by $T_{\epsilon \setminus e}$. Then ϵ is called a *switch* of e and the tree $T_{\epsilon \setminus e}$ is called a *switch*. Moreover, if $e \neq \epsilon$ and $W_{\alpha, \beta}(T_{\epsilon \setminus e})$ is minimum, ϵ is called a best switch (BS) of e w.r.t. $W_{\alpha, \beta}$. Note that $T_{\epsilon \setminus e}$ is a tree, $e \in E(T)$, and $\epsilon \in E(T^c)$.

Now suppose T is an n -vertex and positively weighted tree with coordinate Q_T . We are interested in solving the following problem. Get T , Q_T , $e \in E(T)$ and $w, \alpha, \beta \in \mathbb{R}^{\geq 0}$ and return $T_{\epsilon \setminus e}$, $Q_{T_{\epsilon \setminus e}}$, ϵ and $W_{\alpha, \beta}(T_{\epsilon \setminus e})$, where ϵ is a BS of e w.r.t. $W_{\alpha, \beta}$ and $w = w_{T_{\epsilon \setminus e}}(\epsilon)$. (by $w, \alpha, \beta \in \mathbb{R}^{\geq 0}$ we mean $w > 0, \alpha + \beta > 0, \alpha \geq 0$, and $\beta \geq 0$). We solve the problem with the worst average time of $O(\log n)$ and the best average time of $O(1)$. Clearly, using the output of the problem, if we solve the problem with some complexities, we can keep getting e_i and $w_{i+1}, \alpha_{i+1}, \beta_{i+1} \in \mathbb{R}^{\geq 0}$ and solve the problem on $((T_{\epsilon_1 \setminus e_1})_{\epsilon_2 \setminus e_2} \dots)_{\epsilon_i \setminus e_i}$ with the same complexities on T , $i > 1$. Changing edge weights within the positive range are also supported in $O(1)$ -time per change to update the coordinates $Q_{T_{\epsilon_i \setminus e_i}}$ at any meaningful stage. The mentioned problem is the primary example that will solve in the application section as an application of our tools in the following section. The problem can be seen as a generalization of [7–9] that has found some applications [6,10–12]. For some relevant optimization problems over spanning trees, see [13–

19]. This paper uses extensive notation for more precise interaction, which may take some time to get used to. But our solutions are natural, utilizing some calculus intuition behind the notations.

We might be able to partially compare our solution for the above problem with top tree methods [7–9] (see the final section). Generalizing the top tree method solves the problem above for some specified edges and fixed α, β with the same average complexity as our solution. However, by changing α, β , in the top tree method, one needs to repeat the preprocessing, which is expensive. We also compare our method to the swap edge problem of spanning trees [20,21], which requires a conversion. See the last section for more details.

As the main topics of this paper, we have the following sections. We will define some concepts and achieve some results. Applying the results, we find an efficient solution for BS of positively weighted trees mentioned above. Our method is general, and we have applied it to other problems [22].

2. Discrete Derivative and Integral

Discrete derivative, discrete integral, and convexity are three intuitive concepts for weighted graphs that we define in this section. We have some results and general examples specifically for trees.

Definition 2. Suppose G is a (f_G, w_G) -weighted graph. The discrete derivative of a vertex $a \in V(G)$ toward $x \in N_G(v)$ is defined and denoted as follows:

$$f'_G(\overrightarrow{ax}) = \begin{cases} \frac{f_G(x) - f_G(a)}{d_G(x, a)} & d_G(x, a) \neq 0, \\ f_G(x) - f_G(a) & d_G(x, a) = 0. \end{cases} \quad (1)$$

We say f'_G is consistent if $d_G(x, a) = 0$ results in $f'_G(\overrightarrow{ax}) = 0$, $ax \in E(G)$.

Using Figure 1 as graph G , $f'_G(\overrightarrow{ax}) = \frac{4-3}{3} = \frac{1}{3}$ where $d_G(x, a) = 3$. And $f'_G(\overrightarrow{xa}) = \frac{3-4}{3} = -\frac{1}{3}$.

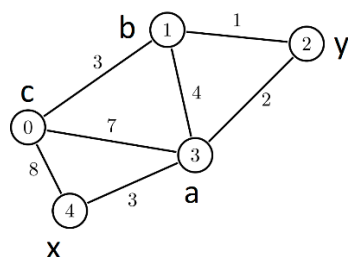


Figure 1. A vertex and edge-weighted graph on 5 vertices.

Lemma 1. Assume f_1 and f_2 are two vertex weight functions for a graph. If $f = \alpha f_1 + \beta f_2$, then $f' = \alpha f'_1 + \beta f'_2$, where $\alpha, \beta \in \mathbb{R}$. Moreover, if f_1 and f_2 are consistent, then f also is.

A path, $v_1 v_2 \dots v_n$, is called f_G -decreasing if it is a path and

$$f_G(v_{i+1}) < f_G(v_i), \quad 0 < i < n.$$

An increasing path is defined in the same sense. Clearly, if $v_1 v_2 \dots v_n$ is a f_G -decreasing path and edge weights are positive, then $f'_G(\overrightarrow{v_i v_{i+1}}) < 0$, $0 < i < n$.

Definition 3. Suppose G is a (f_G, w_G) -weighted graph. Then, $v \in V(G)$ is called f_G -Root or f_G -Roof of G if $f_G(v)$ is minimum or maximum, respectively. Let $A \subset V(G)$ is the set of f_G -

Roots (f_G -Roots) of G . Then we say G is f_G -convex (concave) if and only if for every $v \in V(G)$ there is a decreasing (increasing) path from v to some $a \in A$.

Whenever there is no confusion, we may use Root instead of, f_G -Roots, etc. Using the above image, the vertex with weight 0 is a Root, and the vertex with weight 4 is a Roof in the depicted graph.

To employ discrete derivatives as a tool, we first create some induced weight functions using the current weight of the graph for the vertices of an (f_G, w_G) -weighted graph, G . The first one is as follows:

$$\sigma_G: V(G) \rightarrow \mathbb{R} \quad \text{S.t.} \quad \sigma_G(v) = \sum_{x \in V(G)} d_G(v, x), \quad v \in V(G)$$

Let T be a (σ_T, w_T) -weighted tree (the vertex-weight function is, σ_T). If $ax \in E(T)$ and $w_G(ax) = 0$, then $\sigma'_T(\overrightarrow{ax}) = 0$, otherwise:

$$\sigma'_T(\overrightarrow{ax}) = \frac{\sigma_T(x) - \sigma_T(a)}{d_T(x, a)} = \frac{\sigma_T(x) - \sigma_T(a)}{w(ax)} = n_T(T_a^x) - n_T(T_x^a). \quad (2)$$

Because

$$\sigma_T(x) - \sigma_T(a) = w(ax)[n_T(T_a^x) - n_T(T_x^a)]. \quad (3)$$

Another vertex-weight function for G is induced using the primary weight functions as follows:

$$h_G: V(G) \rightarrow \mathbb{R} \quad \text{S.t.} \quad h_G(v) = \sum_{x \in V(G)} f_G(x) d_G(v, x), \quad v \in G.$$

To help with the computation of h'_T s for a (h_T, w_T) -weighted tree T , let $a \in V(T)$ and x be an a 's neighbor. Thus,

$$\begin{aligned} h_T(a) &= \sum_{v \in V(T_a^x)} f_T(v) d_T(a, v) + \sum_{v \in V(T_x^a)} f_T(v) [d_T(x, v) + w(ax)], \\ h_T(x) &= \sum_{v \in V(T_a^x)} f_T(v) [d_T(a, v) + w(ax)] + \sum_{v \in V(T_x^a)} f_T(v) d_T(x, v). \end{aligned} \quad (4)$$

Using the above equations, $h_T(x) - h_T(a) = w(ax)[f_T(T_a^x) - f_T(T_x^a)]$. Therefore, if $w_T(ax) = 0$, then $h'_T(\overrightarrow{ax}) = 0$, otherwise:

$$h'_T(\overrightarrow{ax}) = \frac{h_T(x) - h_T(a)}{d_T(x, a)} = \frac{w(ax)[f_T(T_a^x) - f_T(T_x^a)]}{w(ax)} = f_T(T_a^x) - f_T(T_x^a). \quad (5)$$

Using Equations (3) and (4), h'_T and σ'_T are consistent for any tree T .

Let $N_T = \{(n_T(T_u^v), n_T(T_v^u))\}_{uv \in E(T)}$ and $F_T = \{(f_T(T_u^v), f_T(T_v^u))\}_{uv \in E(T)}$. Using Equations (2), (5), F_T , and N_T , one can compute σ'_T and h'_T easily. If $f_T(v) = 1$ for every $v \in V(T)$, then $(f_T(T_u^v), f_T(T_v^u)) = ((n_T(T_u^v), n_T(T_v^u)))$ for every $uv \in E(T)$. The following algorithm extracts F_T for a tree T , starts with the edges adjacent to a leaf and then deletes the leaves at each step until the edge set is empty. It is linear regarding the number of vertices and uses the fact that for every $uv \in E(T)$ (Algorithm 1):

$$f_T(T_u^v) + f_T(T_v^u) = f_T(T), \quad f_T(T_v^u) = f_T(v) + \sum_{x \in N_T(v) \setminus u} f_T(T_x^v).$$

Algorithm 1. Computing F_T or N_T

Input: Adjacency list of an n -vertex, nontrivial (f_T, w_T) -weighted tree T and $f_T(T)$

Output: $F_T = \{(f_T(T_u^v), f_T(T_v^u))\}_{uv \in E(T)}$

Initiate with $S = \{a \in V(T) \mid \deg(a) = 1\}$

while $(E(T) \neq \emptyset)$

```

Leaves  $\leftarrow S$ 
 $S = \phi$ 
for  $v \in \text{leaves}$ 
     $u \leftarrow N(v)$ 
     $(f_T(T_u^v), f_T(T_v^u)) \leftarrow (f_T(T) - f_T(v), f_T(v))$ 
     $f_T(u) += f_T(v)$ 
    if  $\deg(u) - 1 = 1$ . #Will be a leaf
         $S = S \cup \{u\}$ 
 $T = T - \{v\}$     #Remove  $v$  from adj list

```

Corollary 1. For an n -vertex tree T , the sets, N_T and, F_T can be computed in $O(n)$.

Corollary 2. Let T be a (f_T, w_T) -weighted tree with n vertices and $g_T = \alpha\sigma_T + \beta h_T$ for some, $\alpha, \beta \in \mathbb{R}$. One can compute and sort $\{g'_T(\overrightarrow{ax}) \mid x \in N_T(a)\}$ for all $a \in V(G)$ with the time complexity of $O(\sum_{v \in V(T)} \deg(v) \log \deg(v))$, that is on average $O(n)$.

The following result gives us an important fact about the weighted trees.

Lemma 2. Suppose $P = v_1 v_2 \dots v_n$, $n > 2$, is a path in a (f_T, w_T) -weighted tree T . If $w_T > 0$ and $\sigma'_T(\overrightarrow{v_1 v_2}) \geq 0$, then $v_2 \dots v_n$ is, σ_T -increasing. If $f_T > 0$, $w_T > 0$, and $h'_T(\overrightarrow{v_1 v_2}) \geq 0$, then $v_2 \dots v_{n+1}$ is, h_T -increasing. If $g_T = \alpha\sigma_T + \beta h_T$, (for $\alpha, \beta \in \mathbb{R}^{\geq 0}, \alpha + \beta \neq 0$) $f_T > 0$, $w_T > 0$, and $g'_T(\overrightarrow{v_1 v_2}) \geq 0$, then $v_2 \dots v_n$ is, g_T -increasing.

Proof. Assume T is a (f_T, w_T) -weighted tree and xyz is a path in T . Without loss of generality, we can prove the lemma for xyz . We know that $V(T_u^v) \cup V(T_v^u) = V(T)$ and $V(T_u^v) \cap V(T_v^u) = \emptyset$ for any $uv \in E(T)$. Thus, for some $A \subset V(T)$,

$$V(T_y^z) = V(T_x^y) \cup \{y\} \cup A, \quad (6)$$

And

$$V(T_z^y) = V(T_y^x) - \{y\} - A. \quad (7)$$

Therefore,

$$V(T_x^y) \subset V(T_y^z) \text{ and } V(T_y^x) \supset V(T_z^y). \quad (8)$$

We use Equations (2) and (5) for computing σ'_T and h'_T . Then by the assumption $\sigma'_T(\overrightarrow{xy}) = n_T(T_x^y) - n_T(T_y^x) \geq 0$. Thus by Equation (8) $\sigma'_T(\overrightarrow{yz}) = n_T(T_y^z) - n_T(T_z^y) > 0$. Since $w_T > 0$, $w(yz)\sigma'_T(\overrightarrow{yz}) > 0$. That is, $\sigma_T(z) > \sigma_T(y)$ which completes the proof for σ_T case. Similarly, assume $h'_T(\overrightarrow{xy}) = f_T(T_x^y) - f_T(T_y^x) \geq 0$. If $f_T > 0$, by Equation (8) $h'_T(\overrightarrow{yz}) = f_T(T_y^z) - f_T(T_z^y) > 0$. If, in addition $w_T > 0$, then $w(yz)h'_T(\overrightarrow{yz}) > 0$. Thus, by Definition 2, $h_T(z) > h_T(y)$. That completes the proof for h_T case. The proof for the g'_T case is like h'_T case. This completes the proof. $\square$

Using Lemma 2, we attained the following result regarding the convexity of a tree.

Theorem 1. Any positively edge-weighted tree is σ -convex, and any positively weighted tree is $(\alpha\sigma + \beta h)$ -convex, for $\alpha, \beta \in \mathbb{R}^{\geq 0}$.

Proof. Let us have a positively edge-weighted tree with at least three vertices. Assume a vertex, v_1 , is not a σ -Root and $v_1 v_2 \dots v_n$ is the shortest possible path (in terms of the number of edges) between v_1 and a σ -Root, v_n . Assume toward a contradiction that $v_1 v_2 \dots v_n$ is not decreasing. Thus, suppose $v_i v_{i+1}$ is the first edge from v_1 toward v_n , such that $\sigma(v_i) \leq \sigma(v_{i+1})$. Since the edge weights are positive, $\sigma'(\overrightarrow{v_i v_{i+1}}) \geq 0$. Then, by Lemma 2, the path $v_{i+1} v_{i+2} \dots v_n$ is increasing. That is, $\sigma(v_i) \leq \sigma(v_n)$ and $i < n$. If

$\sigma(v_i) = \sigma(v_n)$, v_i also is a σ -Root and $v_1 v_2 \dots v_i$ is shorter than $v_1 v_2 \dots v_n$ while having the same specification, which is a contradiction with $v_1 v_2 \dots v_n$ being the shortest. And if, $\sigma(v_i) < \sigma(v_n)$, contradicts with, v_n being a σ -Root. Therefore, $v_1 v_2 \dots v_n$ is decreasing. Since v_1 was arbitrary, the tree is σ -convex. For the proof of $\alpha\sigma + \beta h$ case, use the same strategy as for σ , along with the positivity of the vertices' weight. $\square$

Corollary 3. Assume that $g = \alpha\sigma + \beta h$, $\alpha, \beta \in \mathbb{R}^{\geq 0}$, $\alpha + \beta \neq 0$. Then, a positively weighted tree has at most two g -Roots. If x and y are two g -Roots of a tree, then they are adjacent, and $g'(\overrightarrow{xy}) = g'(\overrightarrow{yx}) = 0$. Moreover, if c is a tree's g -Root and v is a non-Root neighbor of c , then $g'(\overrightarrow{cv}) > 0$, and $g'(\overrightarrow{vc}) < 0$. Usefully, if $P = v_1 v_2 \dots v_m$ is a decreasing path between v_1 and a g -Root, v_m , then $g'(\overrightarrow{v_i a}) < 0$ if and only if $a = v_{i+1}$, $i < m$. In addition, P is the longest g -decreasing path starting from v_1 . Finally, a leaf is not a $(\alpha\sigma + \beta h)$ -Root of a positively weighted tree with more than 2 vertices.

Definition 4. Suppose G is a (f_G, w_G) -weighted graph. Regarding f'_G and d_G , we define and denote the discrete integral of f'_G along a path, $P = v_1 v_2 \dots v_n$ from v_1 to v_n as follows:

$$\int_P f'_G d_G = \sum_{i=1}^{n-1} f'_G(\overrightarrow{v_i v_{i+1}}) \cdot d_G(v_i, v_{i+1}).$$

If $a, b \in V(G)$ and $P_G(a, b)$ is an arbitrary path between a and b , we denote $\int_{P_G(a, b)} f'_G d_G$ by

$$\int_a^b f'_G d_G$$

By Definitions 2 and 4, we have:

Theorem 2. Suppose G is a connected (f_G, w_G) -weighted graph and $a, b \in V(G)$. If f'_G is consistent, then for any path from a to b ,

$$\int_a^b f'_G d_G = f_G(b) - f_G(a).$$

Using Figure 1 as graph G with the given weights and the above result or Definition 2.12, one can see that,

$$\begin{aligned} \int_x^y f'_G d_G &= \int_{P=xay} f'_G d_G = \int_{P=xaby} f'_G d_G = \int_{P=xacby} f'_G d_G = \int_{P=xcbay} f'_G d_G \\ &= \int_{P=xacay} f'_G d_G = \int_{P=xcabay} f'_G d_G = -2. \end{aligned}$$

3. Some Applications of Discrete Integral and Derivative and Convexity

In this section, we formally define three optimization problems on trees. As an application of the results of the previous section, we will have efficient solutions to the problems with some possible comparisons with existing solutions at the end. The compared problems are not clearly defined in the same way we do. We try to convert them for some partial comparison.

The *routing cost* of G for a given set of sources $S \subseteq V(G)$ and the total distance of G are defined as follows, respectively:

$$RC(G, S) = \sum_{v \in V(G)} \sum_{s \in S} d_G(v, s), \quad D(G) = \frac{1}{2} \sum_{v \in V(G)} \sum_{s \in V(G)} d_G(v, s).$$

The Wiener index, $W(G)$, of a simple graph G , ($f_G = 1, w_G = 1$) is equal to its total distance [23,24]. By setting α, β, w_G , and f_G ; $W_{\alpha, \beta}$, defined in the introduction, can produce any of $D, W, W_+, W_\times$, or RC for a graph, G . So, $W_{\alpha, \beta}$ generalizes all the mentioned graph invariants.

Definition 5. Let T be a weighted tree and $T_{\epsilon \setminus e} = T - e + \epsilon$, where $e \in E(T)$ and $\epsilon \in E(T^c)$.

1. We say $\epsilon \in E(T^c)$ is a switch of $e \in E(T)$, if $T_{\epsilon \setminus e}$ is a tree. Then $T_{\epsilon \setminus e}$ is called a switch.
2. We say $\epsilon \in E(T^c)$ is a BS of $e \in E(T)$ w.r.t. $W_{\alpha, \beta}$ if $W_{\alpha, \beta}(T_{\epsilon \setminus e})$ is minimum, i.e. $W_{\alpha, \beta}(T_{\epsilon \setminus e}) = \min_{s \in E(T^c)} W_{\alpha, \beta}(T_{s \setminus e})$.
3. The coordinate of T is shown and defined with $Q_T = \{F_T, N_T, W_{\times}(T), W_{+}(T)\}$.

Based on the BS definition, we define three problems. The second and third problems are created merely for comparison. Assume T is a weighted tree with $E(T) = \{e_i\}_{i=1}^n$ and we have coordinated, Q_T .

Problem1:

Input:

1. T ,
2. Q_T ,
3. $e \in E(T)$,
4. $w, \alpha, \beta \in \mathbb{R}^{\geq 0}$.

Output:

1. $\epsilon \in E(T^c)$, where ϵ is a BS of e w.r.t. $W_{\alpha, \beta}$ with $w_{T_{\epsilon \setminus e}}(\epsilon) = w$.
2. $W_{\alpha, \beta}(T_{\epsilon \setminus e})$,
3. $T_{\epsilon \setminus e}$,
4. $Q_{T_{\epsilon \setminus e}}$.

Problem2:

Input:

1. T with $E(T) = \{e_i\}_{i=1}^n$
2. Q_T ,
3. $\{w_i, \alpha_i, \beta_i \in \mathbb{R}^{\geq 0}\}_{i=1}^n$.

Output:

1. r ,
2. $\epsilon_r \in E(T^c)$,
3. $T_{\epsilon_r \setminus e_r}$,
4. $Q_{T_{\epsilon_r \setminus e_r}}$,
5. $W_{\alpha_r, \beta_r}(T_{\epsilon_r \setminus e_r})$,

where $W_{\alpha_r, \beta_r}(T_{\epsilon_r \setminus e_r}) = \min\{W_{\alpha_i, \beta_i}(T_{\epsilon_i \setminus e_i})\}_{i=1}^n$ and ϵ_i is a BS of e_i w.r.t. W_{α_i, β_i} with $w_i = w_{T_{\epsilon_i \setminus e_i}}(\epsilon_i)$. (Overall best switch).

Problem3:

Input:

1. T with $E(T) = \{e_i\}_{i=1}^n$,
2. Q_T ,
3. $\{w_i, \alpha_i, \beta_i \in \mathbb{R}^{\geq 0}\}_{i=1}^n$.

Output:

1. $\{(\epsilon_i \in E(T^c), W_{\alpha_i, \beta_i}(T_{\epsilon_i \setminus e_i}))\}_{i=1}^n$,

where ϵ_i is a BS of e_i w.r.t. W_{α_i, β_i} with $w_i = w_{T_{\epsilon_i \setminus e_i}}(\epsilon_i)$. (All best switches).

Definition 6. For a numerical graph invariant, W_* , and a switch $T_{s \setminus t}$ let

$$W'_*(T_{s \setminus t}) = W_*(T_{s \setminus t}) - W_*(T).$$

Lemma 3. Let T be a (f_T, w_T) -weighted tree, $xy \in E(T)$, and $g_T = \alpha \sigma_T + \beta h_T$ for some $\alpha, \beta \in \mathbb{R}$. Using, F_T and N_T , we can calculate $g'_{T_{xy}}$ on the path from x toward any vertex of T_x^y . A similar argument holds for a path from y toward any vertex of T_y^x .

Proof: Let T be a (f_T, w_T) -weighted tree and $xy \in E(T)$ with $T - \{xy\} = T_x^y \cup T_y^x$ and $g_T = \alpha\sigma_T + \beta h_T$, $\alpha, \beta \in \mathbb{R}$. Assume $v_1 v_2 \dots v_n$ is a path in T_x^y , with $v_1 = x$. Using Equations (1), (2), (5), and $\{N_T, F_T\} = \{(n_T(T_u^v), n_T(T_v^u)), (f_T(T_u^v), f_T(T_v^u))\}_{uv \in E(T)}$ we could extract, σ'_T and h'_T , and so $g'_T = \alpha\sigma'_T + \beta h'_T$ on $v_1 v_2 \dots v_n$. However, we need to get $g'_{T_x^y}$ through $v_1 v_2 \dots v_n$. To do so, one can check for $v_1 v_2 \dots v_n$,

$$(n_{T_x^y}([T_x^y]_{v_i}^{v_{i+1}}), n_{T_x^y}([T_x^y]_{v_{i+1}}^{v_i})) = (n_T(T_{v_i}^{v_{i+1}}) - n_T(T_y^x), n_T(T_{v_{i+1}}^{v_i})), \quad 0 < i < n, \quad (9)$$

$$(f_{T_x^y}([T_x^y]_{v_i}^{v_{i+1}}), f_{T_x^y}([T_x^y]_{v_{i+1}}^{v_i})) = (f_T(T_{v_i}^{v_{i+1}}) - f_T(T_y^x), f_T(T_{v_{i+1}}^{v_i})). \quad 0 < i < n. \quad (10)$$

Thus, using Equations (1), (2) and (5), along with (9) and (10):

$$\sigma'_{T_x^y}(\overrightarrow{v_i v_{i+1}}) = \sigma'_T(\overrightarrow{v_i v_{i+1}}) - n_T(T_y^x), \quad h'_{T_x^y}(\overrightarrow{v_i v_{i+1}}) = h'_T(\overrightarrow{v_i v_{i+1}}) - f_T(T_y^x).$$

That is, using F_T and N_T , we can compute $g'_{T_x^y}(\overrightarrow{v_i v_{i+1}})$ as follows:

$$g'_{T_x^y}(\overrightarrow{v_i v_{i+1}}) = \alpha\sigma'_T(\overrightarrow{v_i v_{i+1}}) + \beta h'_T(\overrightarrow{v_i v_{i+1}}) - \alpha n_T(T_y^x) - \beta f_T(T_y^x).$$

Symmetrically, the argument holds for the T_y^x and y case. $\square$

It is straightforward but involves some long computation to find out the following.

Lemma 4. Suppose T is a (f_T, w_T) -weighted tree with $uv \in E(T)$ and $xy \in E(T^c)$. If $x \in V(T_u^v)$ and $y \in V(T_v^u)$, then

$$W'_{\alpha, \beta}(T_{xy \setminus uv}) = \int_u^x p' d_{T_u^v} + \int_v^y q' d_{T_v^u} + FN_{\alpha, \beta}(uv)[w_{T_{xy \setminus uv}}(xy) - w_T(uv)],$$

where,

$$FN_{\alpha, \beta}(uv) = [\alpha f_T(T_u^v) f_T(T_v^u) + \beta (n_T(T_u^v) f_T(T_v^u) + f_T(T_u^v) n_T(T_v^u))], \quad \alpha, \beta \in \mathbb{R},$$

and

$$p(a) = [\alpha f_T(T_v^u) + \beta n_T(T_v^u)] h_{T_u^v}(a) + \beta f_T(T_v^u) \sigma_{T_u^v}(a), \quad a \in V(T_u^v),$$

$$q(b) = [\alpha f_T(T_u^v) + \beta n_T(T_u^v)] h_{T_v^u}(b) + \beta f_T(T_u^v) \sigma_{T_v^u}(b), \quad b \in V(T_v^u).$$

Proof. Assume T is a tree. If we remove $uv \in E(T)$ and connect, T_u^v and T_v^u with $xy \in E(T^c)$, where $x \in V(T_u^v)$ and $y \in V(T_v^u)$, then

$$\begin{aligned} W_{\alpha, \beta}(T_{xy \setminus uv}) &= W_{\alpha, \beta}(T_u^v) + W_{\alpha, \beta}(T_v^u) \\ &\quad + \sum_{a \in V(T_u^v)} \sum_{b \in V(T_v^u)} [\alpha f_T(a) f_T(b) + \beta (f_T(a) + f_T(b))] \cdot [d_T(x, a) \\ &\quad + w_{T_{xy \setminus uv}}(xy) + d_T(b, y)] \\ &= W_{\alpha, \beta}(T_u^v) + W_{\alpha, \beta}(T_v^u) + [\alpha f_T(T_v^u) + \beta n_T(T_v^u)] h_{T_u^v}(x) + [\alpha f_T(T_u^v) + \beta n_T(T_u^v)] h_{T_v^u}(y) \\ &\quad + \beta [f_T(T_v^u) \sigma_{T_u^v}(x) + f_T(T_u^v) \sigma_{T_v^u}(y) \\ &\quad + w_{T_{xy \setminus uv}}(xy) [\alpha f_T(T_u^v) f_T(T_v^u) \\ &\quad + \beta (n_T(T_u^v) f_T(T_v^u) + f_T(T_v^u) f_T(T_u^v))] \end{aligned} \quad (11)$$

Using the fact that $W_x(T_{uv \setminus uv}) = W_x(T)$ and the above formula,

$$\begin{aligned} W'_{\alpha, \beta}(T_{xy \setminus uv}) &= W_{\alpha, \beta}(T_{xy \setminus uv}) - W_{\alpha, \beta}(T) \\ &= [p(x) - p(u)] + [q(y) - q(v)] + FN_{\alpha, \beta}(uv)[w_{T_{xy \setminus uv}}(xy) \\ &\quad - w_T(uv)]. \end{aligned}$$

where $FN_{\alpha, \beta}(uv) = [\alpha f_T(T_u^v) f_T(T_v^u) + \beta (n_T(T_u^v) f_T(T_v^u) + f_T(T_u^v) n_T(T_v^u))]$ and

$$p(a) = [\alpha f_T(T_v^u) + \beta n_T(T_v^u)]h_{T_u^v}(a) + \beta f_T(T_v^u)\sigma_{T_u^v}(a), \quad a \in V(T_u^v),$$

$$q(b) = [\alpha f_T(T_u^v) + \beta n_T(T_u^v)]h_{T_v^u}(b) + \beta f_T(T_u^v)\sigma_{T_v^u}(b), \quad b \in V(T_v^u).$$

Since T_u^v and T_v^u are connected, and p and q are consistent; by Lemma 1 and Theorem 2, we have:

$$W'_{\alpha,\beta}(T_{xy \setminus uv}) = \int_u^x p' d_{T_u^v} + \int_v^y q' d_{T_v^u} + FN_{\alpha,\beta}(uv) [w_{T_{xy \setminus uv}}(xy) - w_T(uv)]. \square$$

The above lemma gives a clue between a BS of $uv \in E(T)$ w.r.t $W_{\alpha,\beta}$ and the p -Roots and q -Roots of T_u^v and T_v^u . Often, finding a BS is equivalent to finding some Roots.

Lemma 5. Suppose T is a positively weighted tree with at least three vertices and $uv \in E(T)$. If a and b are a p -Root of T_u^v and a q -Root of T_v^u , respectively, then a BS of uv w.r.t $W_{\alpha,\beta}$ is xy with $x \in N_T(a) \cup \{a\}$ and $y \in N_T(b) \cup \{b\}$. More precisely, let the following be vertex weight functions for the relevant vertices:

$$p(c) = [\alpha f_T(T_v^u) + \beta n_T(T_v^u)]h_{T_u^v}(c) + \beta f_T(T_v^u)\sigma_{T_u^v}(c), \quad c \in V(T_u^v),$$

$$q(c) = [\alpha f_T(T_u^v) + \beta n_T(T_u^v)]h_{T_v^u}(c) + \beta f_T(T_u^v)\sigma_{T_v^u}(c), \quad c \in V(T_v^u).$$

Then a choice of xy as a BS of uv w.r.t $W_{\alpha,\beta}$ is as follows:

- A** If $(u, v) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$,
1. $(x, y) = (u, z)$ where $q(z) = \min_{c \in N_{T_v^u}(v)} q(c)$, or
 2. $(x, y) = (z, v)$ where $p(z) = \min_{c \in N_{T_u^v}(u)} p(c)$.

If $q'(\overrightarrow{vz})w_T(vz) < p'(\overrightarrow{uz})w_T(uz)$, 1 gives a choice for xy ; otherwise, 2.

- B** If $(u, v) \neq (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$, then $(x, y) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$.

Moreover, $F_T = F_{T_{xy \setminus uv}}$ and $N_T = N_{T_{xy \setminus uv}}$ except for the edges on $P_T(u, x)$ and $P_T(v, y)$. More precisely, if $P_{T_u^v}(u, x) = u_1 u_2 \dots u_m$ is the path between $u = u_1$ and $x = u_m$,

$$(n_{T_{xy \setminus uv}}(T_{u_i}^{u_{i+1}}), n_{T_{xy \setminus uv}}(T_{u_{i+1}}^{u_i})) = (n_T(T_{u_i}^{u_{i+1}}) - n_T(T_v^u), n_T(T_{u_{i+1}}^{u_i}) + n_T(T_v^u)),$$

$$(f_{T_{xy \setminus uv}}(T_{u_i}^{u_{i+1}}), f_{T_{xy \setminus uv}}(T_{u_{i+1}}^{u_i})) = (f_T(T_{u_i}^{u_{i+1}}) - f_T(T_v^u), f_T(T_{u_{i+1}}^{u_i}) + f_T(T_v^u)), \quad 0 < i < m.$$

And, if $P_{T_v^u}(v, y) = v_1 v_2 \dots v_n$ is a path between $v = v_1$ and $y = v_n$,

$$(n_{T_{xy \setminus uv}}(T_{v_i}^{v_{i+1}}), n_{T_{xy \setminus uv}}(T_{v_{i+1}}^{v_i})) = (n_T(T_{v_i}^{v_{i+1}}) - n_T(T_u^v), n_T(T_{v_{i+1}}^{v_i}) + n_T(T_u^v)),$$

$$(f_{T_{xy \setminus uv}}(T_{v_i}^{v_{i+1}}), f_{T_{xy \setminus uv}}(T_{v_{i+1}}^{v_i})) = (f_T(T_{v_i}^{v_{i+1}}) - f_T(T_u^v), f_T(T_{v_{i+1}}^{v_i}) + f_T(T_u^v)), \quad 0 < i < n.$$

Proof. Assume we have the lemma assumptions. By Equation (11),

$$W_{\alpha,\beta}(T_{st \setminus uv}) = p(s) + q(t) + r, \quad s \in V(T_u^v) \text{ and } t \in V(T_v^u). \quad (12)$$

where r is independent of the choice of s and t . And the values of $p(s)$ and $q(t)$ are independent of each other. By definition, if $a \in V(T_u^v)$ and $b \in V(T_v^u)$ minimize Equation (12) and $ab \neq uv$, then ab is a BS of uv . One sees that by minimizing Equation (12) inclusively, we do not exclude uv as a choice for ab , since:

$$\min_{s \in V(T_u^v)} p(s) + \min_{t \in V(T_v^u)} q(t) + r = \min_{st \in E(T^c) \cup \{uv\}} W_{\alpha,\beta}(T_{st \setminus uv}). \quad (13)$$

Accordingly, (a, b) minimizes Equation (12) if and only if $(p(a), q(b)) = (\min_{s \in V(T_u^v)} p(s), \min_{t \in V(T_v^u)} q(t))$. That is, $(a, b) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$.

If case **A** happens, which is $(u, v) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$ and also $(x, y) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$, then we might have $xy = uv$. To exclude uv in the minimization of Equation (12), we avoid the case that u and v appear concurrently by letting $(p(x), q(y))$ is equal to either $(\min_{s \in V(T_u^v) \setminus u} p(s), \min_{t \in V(T_v^u)} q(t))$ or $(\min_{s \in V(T_u^v)} p(s), \min_{t \in V(T_v^u) \setminus v} q(t))$. By Theorem 1 and the assumptions T_u^v and T_v^u are convex. That is,

$$\min_{s \in V(T_u^v) \setminus u} p(s) = \min_{s \in N_{T_u^v}(u)} p(s) \text{ and } \min_{t \in V(T_v^u) \setminus v} q(t) = \min_{t \in N_{T_v^u}(v)} q(t).$$

As a result, either

$$1\text{-(}x, y) = (u, z) \text{ where } q(z) = \min_{b \in N_{T_v^u}(u)} q(b), \text{ or}$$

$$2\text{-(}x, y) = (z, v) \text{ where } p(z) = \min_{a \in N_{T_u^v}(v)} p(a).$$

Per Lemma 4, in case 1—, $W'_{\alpha, \beta}(T_{xy \setminus uv}) - r = \int_u^{u=x} p' d_{T_u^v} + \int_v^y q' d_{T_v^u} = \int_v^y q' d_{T_v^u} = q'(\overline{vz})w_T(vz)$ and in case 2— $W'_{\alpha, \beta}(T_{xy \setminus uv}) - r = p'(\overline{uz})w_T(uz)$. Thus, if $p'(\overline{uz})w_T(uz) > q'(\overline{vz})w_T(vz)$, then 1 gives the choice of (x, y) . If $p'(\overline{uz})w_T(uz) < q'(\overline{vz})w_T(vz)$, then (x, y) comes from 2. Otherwise, either 1 or 2 is the choice.

Now assume $(u, v) \neq (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$. If $(a, b) = (p\text{-Root of } T_u^v, q\text{-Root of } T_v^u)$, then $(u, v) \neq (a, b)$. Moreover, (a, b) minimizes $W_{\alpha, \beta}$ in Equation (12). So $(x, y) = (a, b)$ is the right choice. This completes the proof of **B**.

To form $T_{xy \setminus uv}$ we remove $uv \in E(T)$ and reconnect the components of $T - u$ using $x \in V(T_u^v)$ and $y \in V(T_v^u)$, $uv \neq xy$. Thus, $F_T \neq F_{T_{xy \setminus uv}}$ and $N_T \neq N_{T_{xy \setminus uv}}$ in general. Precisely, $(n_T(T_a^b), n_T(T_b^a)) \neq (n_{T_{xy \setminus uv}}(T_a^b), n_{T_{xy \setminus uv}}(T_b^a))$ or $(f_T(T_a^b), f_T(T_b^a)) \neq (f_{T_{xy \setminus uv}}(T_a^b), f_{T_{xy \setminus uv}}(T_b^a))$ happens if $ab \in P_T(u, x)$ or $ab \in P_T(v, y)$. By Definition 1 $f_T(T_a^b) + f_T(T_b^a) = f_T(T)$ and $n_T(T_a^b) + n_T(T_b^a) = |V(T)|$. And, by Lemma 3 Equations (9) and (10), if $P_{T_u^v}(u, x) = u_1 u_2 \dots u_m$ is the path between $u = u_1$ and $x = u_m$,

$$(n_{T_{xy \setminus uv}}(T_{u_i}^{u_{i+1}}), n_{T_{xy \setminus uv}}(T_{u_{i+1}}^{u_i})) = (n_T(T_{u_i}^{u_{i+1}}) - n_T(T_v^u), n_T(T_{u_{i+1}}^{u_i}) + n_T(T_v^u)),$$

$$(f_{T_{xy \setminus uv}}(T_{u_i}^{u_{i+1}}), f_{T_{xy \setminus uv}}(T_{u_{i+1}}^{u_i})) = (f_T(T_{u_i}^{u_{i+1}}) - f_T(T_v^u), f_T(T_{u_{i+1}}^{u_i}) + f_T(T_v^u)), 0 < i < m.$$

And symmetrically, for $P_{T_v^u}(v, y) = v_1 v_2 \dots v_n$ with $v = v_1$ and $y = v_n$,

$$(n_{T_{xy \setminus uv}}(T_{v_i}^{v_{i+1}}), n_{T_{xy \setminus uv}}(T_{v_{i+1}}^{v_i})) = (n_T(T_{v_i}^{v_{i+1}}) - n_T(T_u^v), n_T(T_{v_{i+1}}^{v_i}) + n_T(T_u^v)),$$

$$(f_{T_{xy \setminus uv}}(T_{v_i}^{v_{i+1}}), f_{T_{xy \setminus uv}}(T_{v_{i+1}}^{v_i})) = (f_T(T_{v_i}^{v_{i+1}}) - f_T(T_u^v), f_T(T_{v_{i+1}}^{v_i}) + f_T(T_u^v)), 0 < i < n.$$

□

Lemma 6. For a (f_T, w_T) -weighted tree T ,

$$W_{\times}(T) = \sum_{uv \in E(T)} [f_T(T_u^v) f_T(T_v^u)] w_T(uv),$$

and,

$$W_{+}(T) = \sum_{uv \in E(T)} [n_T(T_u^v) f_T(T_v^u) + n_T(T_v^u) f_T(T_u^v)] w_T(uv).$$

Proof. We prove the first equation using the double-counting principle. Assume T' is a copy of T with the edge weights set to 0 at the start. Let $u, v \in V(T)$ and $P_T(u, v)$ is the path between u and v . For every $ab \in P_T(u, v)$ add the number $w(uv)f_T(a)f_T(b)$ to

the corresponding edge of ab in T' . Then $\sum_{uv \in P_T(u,v)} w_{T'}(uv) = [f_T(u) f_T(v)] d_T(u, v)$. Thus, suppose we repeat the above process for every pair $u, v \in V(T)$. If so, then

$$\sum_{uv \in E(T')} w_{T'}(uv) = \sum_{\{u,v\} \subseteq V(T)} [f_T(u) \times f_T(v)] d_T(u, v) = W_{\times}(T).$$

And on the other hand, one can see that $w_{T'}(uv) = w_T(uv)_T [f_T(T_u^v) f_T(T_v^u)]$ for $uv \in E(T')$. Thus, $W_{\times}(T) = \sum_{uv \in E(T')} w_{T'}(uv) = \sum_{uv \in E(T)} w_T(uv) [f_T(T_u^v) f_T(T_v^u)]$. That proves the formula of $W_{\times}(T)$. The proof for the W_+ case is quite similar. $\square$

Corollary 4. Assume T is a (f_T, w_T) -weighted n -vertex tree. We can compute $W_{\times}(T)$ and $W_+(T)$ in $O(n)$ -time. Moreover, if by changing the weight of $ab \in E(T)$ to w , we get a tree T' , then

$$W_{\times}(T') = W_{\times}(T) + (w - w_T(ab)) f_T(T_u^v) f_T(T_v^u),$$

$$W_+(T') = W_+(T) + (w - w_T(ab)) [n_T(T_u^v) f_T(T_v^u) + n_T(T_v^u) f_T(T_u^v)]$$

Remark 1. Here, we give a simple intuition behind an algorithm for problem 1 when the tree T is positively weighted. Using Lemma 5, finding a BS for $uv \in E(T)$, we need to have the p -Roots and q -Roots of T_u^v and T_v^u , respectively. By definition $u \in V(T_u^v)$ and $v \in V(T_v^u)$. By Corollary 3, the longest decreasing path from u ends up with a p -Root of T_u^v say x , and similarly, the longest decreasing path from v ends up at a q -Root of T_v^u say y . Moreover, by Corollary 3, those paths are the only decreasing paths in their respective components. So, one can easily take some derivatives to reach from u to x or from v to y since the paths are decreasing and unique. Then by Lemma 5, a BS of uv is one of the following cases as detailed in the lemma.

1. xy ,
2. an edge between x and a vertex of $N_T(y)$,
3. an edge between y and a vertex of $N_T(x)$.

Lemma 5 details how to choose among the above limited cases using derivatives. As we find the unique and decreasing path between u and x and also v and y , one can update Q_T as in Lemma 5. Having those paths also eases calculating $W_{\times}$, W_+ , and $W_{\alpha,\beta}$ from Lemma 4. The following proposition details the mentioned intuition with proof of time complexity.

Proposition 1. Let T be an n -vertex positively weighted tree with a coordinate, Q_T . If we get $e \in E(T)$ and $w, \alpha, \beta \in \mathbb{R}^{\geq 0}$, then we can compute ϵ , $T_{\epsilon \setminus e}$, $Q_{T_{\epsilon \setminus e}}$, and $W_{\alpha,\beta}(T_{\epsilon \setminus e})$, with the worst average time of $O(\log n)$ and the best average time of $O(1)$, where ϵ is a BS of e w.r.t $W_{\alpha,\beta}$ and $w_{T_{\epsilon \setminus e}}(\epsilon) = w$. Updating, Q_T after edge-weight change to a positive number is supported in $O(1)$ -time per change.

Proof. Assume we have an n -vertex positively weighted tree T , $Q_T = \{F_T, N_T, W_{\times}(T), W_+(T)\}$ and $uv \in E(G)$, $n > 2$. Let p and q be some induced vertex weight functions for T_u^v and T_v^u as follows,

$$p_{\alpha,\beta}(c) = [\alpha f_T(T_v^u) + \beta n_T(T_v^u)] h_{T_u^v}(c) + \beta f_T(T_v^u) \sigma_{T_u^v}(c), \quad c \in V(T_u^v),$$

$$q_{\alpha,\beta}(c) = [\alpha f_T(T_u^v) + \beta n_T(T_u^v)] h_{T_v^u}(c) + \beta f_T(T_u^v) \sigma_{T_v^u}(c), \quad c \in V(T_v^u).$$

Then with the results so far, we have the following:

1. Computing a p' or q' in $O(1)$ -time. Using F_T and N_T , Lemma 3, and Definition 2, computing $p'(\overrightarrow{rs})$ costs $O(1)$ -time for any $rs \in P_{T_u^v}(u, x)$ where $x \in V(T_u^v)$ and we know the direction. A similar argument holds for computing q' on $P_{T_v^u}(v, y)$.

2. Checking whether a vertex is a Root on average of $O(1)$ -time. By Corollary 3, if $p'(\overrightarrow{ux}) < 0$ for only one $x \in N_{T_u^v}(u)$, then u is not a p -Root. The following can verify whether a vertex is p or q -Root or not. (Let $f = p$ or q)

Takes a vertex and a f and says it is a f -Root or not

```

IS_Root( $a, f$ )
For  $s \in N(a)$ 
  if  $f'(\overrightarrow{as}) < 0$ 
    return False
  return True

```

The average degree in a tree is $O(1)$. Thus, **IS_Root** run time is $O(1)$ on average.

3. Finding a neighbor with a minimum p or q in $O(1)$ -time on average. If $s \in N_{T_u^v}(a)$, then by Theorem 2 and Definition 4, $p(s) = p'(\overrightarrow{as})w(as) + p(a)$. By **1**- computing $p(s)$ takes $O(1)$ -time. Moreover, $O(|N_T(a)|) = O(1)$ on average. Thus, finding x with $p(x) = \min_{s \in N_{T_u^v}(a)} p(s)$ takes $O(1)$ -time on average. Computing min q neighbor follows the same rules. One implements the mentioned idea as follows.

Takes a vertex, a , finds $x \in N(a)$ with min f and returns x and ax ($f = p$ or q)

```

FND_min( $a, f$ )
 $min = \infty$ 
For  $s \in N(a)$ 
   $f(s) = f'(\overrightarrow{as})w(as) + f(a)$ 
  if  $f(s) < min$ 
     $min = f(s)$ 
     $x \leftarrow s$ 

return  $x, ax$ 

```

4. Finding, $P_{T_u^v}(u, x)$ or $P_{T_v^u}(v, y)$ in $O(\log n)$ on average, where x and y are p and q -Root. By Theorem 1, T_u^v is p -convex. Assume that $P = u_1 u_2 \dots u_m$ is the decreasing path between $u = u_1$ and a p -Root, u_m . By Corollary 3, P is the longest decreasing path beginning with u , and $p'(\overrightarrow{u_i a}) < 0$ if and only if $a = u_{i+1}$, $a \in N(v_i)$, $0 < i < m$. So, starting with $i = 1$, we can find u_{i+1} when we have u_i and compute P as follows:

Takes a vertex, a , and returns c and $P(a, c)$ where c is a f ($f = p$ or q)

```

Root_Path( $a, f$ )
 $P(a, c) = a_1 = a$ 
Lable
for  $a_{i+1} \in N(a_i) \setminus a_{i-1}$ 
  if  $f'(\overrightarrow{a_i a_{i+1}}) < 0$ .
     $P(a, c) = P(a, c) \cup a_{i+1}$ 
     $a_i \leftarrow a_{i+1}$ 
  Break and Goto Lable
return  $a, P(a, c)$ 

```

For finding u_{i+1} from u_i , we can exclude u_{i-1} and jump to u_{i+1} as the process **Root_Path** does. That reduces the average complexity. Moreover, when we are on the vertex u_i , $i < m$, one takes at most $|N(u_i)|$ derivatives to find u_{i+1} , since we know, u_{i+1} is the only vertex with negative derivatives toward it. So, **Root_Path**(u, p) returns, $P_{T_u^v}(u, x)$ by taking some derivatives, at most, as many as the total degree of the vertices on $P_{T_u^v}(u, x)$, where x is a p -Root. By [25], the path length in a tree is $O(\log n)$ on average, where it can be $O(1)$ as well. In addition, the average degree in a tree is $O(1)$, and taking each derivative requires $O(1)$ -time. Moreover, x is a p -Root (has min p) and $T_u^v < T$. Thus, the worst average run time of **Root_Path**(u, p) is $O(\log n)$ and the best average time is $O(1)$. Similarly, **RCE_Path**(v, q) has the same run time to return $P_{T_v^u}(v, y)$ where y is a q -Root.

5. Computing $\int_u^x p' d_{T_u^v}$ and $\int_v^y q' d_{T_v^u}$ in $O(|P(u, x)|)$ -time and $O(|P(v, y)|)$ -time, respectively. Per Definition 4, Theorem 2, and 1, if we have a path, $P = a_1 a_2 \dots a_m$, we can compute the integral of P in $O(m)$ -time as follows:

Takes a path P and f and returns $\int_p f'$ ($f = p$ or q)

```

Integ ( $P = a_1 a_2 \dots a_m, f$ )

```

```

 $\int_{a_1}^{a_m} f' d = 0$ 
for  $0 < i < m$ 
     $\int_{a_1}^{a_m} f' d += f'(\overrightarrow{a_i a_{i+1}}) \cdot w(a_i a_{i+1})$ 
return  $\int_{a_1}^{a_m} f' d$ 

```

6. Updating F_T to $F_{T_{xy \setminus uv}}$ and N_T to $N_{T_{xy \setminus uv}}$ in $O(|P(u, x)|) + O(|P(v, y)|)$ time.
 Per Lemma 5, $\mathbf{Up_FN}(P(u, x), f_T(T_v^u), n_T^u(v))$ and $\mathbf{Up_FN}(P(v, y), f_T(T_u^v), n_T^v(u))$ updates F_T to $F_{T_{xy \setminus uv}}$ and N_T to $N_{T_{xy \setminus uv}}$, where $\mathbf{Up_FN}$ is a process as follows.

```

# Takes a path and updates  $F_T$  and  $N_T$  accordingly
 $\mathbf{Up\_FN}(P = v_1 v_2 \dots v_m, F, N)$ 
for  $0 < i < m$ 
     $(n_T(T_{v_i}^{v_{i+1}}), n_T(T_{v_{i+1}}^{v_i})) \leftarrow (n_T(T_{v_i}^{v_{i+1}}) - N, n_T(T_{v_{i+1}}^{v_i}) + N)$ 
     $(f_T(T_{v_i}^{v_{i+1}}), f_T(T_{v_{i+1}}^{v_i})) \leftarrow (f_T(T_{v_i}^{v_{i+1}}) - F, f_T(T_{v_{i+1}}^{v_i}) + F)$ 

```

Using the devised procedures in 1- to 6-, we can implement Lemma 4 and 5, which finds us ϵ , $T_{\epsilon \setminus e}$, $Q_{T_{\epsilon \setminus e}}$, and $W_{\alpha, \beta}(T_{\epsilon \setminus e})$, for the input $e \in E(T)$ and $w, \alpha, \beta \in \mathbb{R}^{\geq 0}$, where ϵ is a BS of e w.r.t $W_{\alpha, \beta}$ and $w_{T_{\epsilon \setminus e}}(\epsilon) = w$. The full detail is presented with the following algorithm: BS algorithm (Algorithm 2). Each line's average time complexity bound is shown in the results.

Preprocessing-----

Compute $Q_T = \{F_T, N_T, W_{\times}(T), W_{+}(T)\}$ # $O(n)$ Corollary 1 & Lemma 6

Algorithm 2. BS Algorithm

Input: Adj list of a (f_T, w_T) -weighted tree T , $uv \in E(T)$, Q_T , $w, \alpha, \beta \in \mathbb{R}^+$

Output: xy , $T_{xy \setminus uv}$, $W_{\alpha, \beta}(T_{xy \setminus uv})$, $Q_{T_{xy \setminus uv}}$, where xy is a BS of uv w.r.t. $W_{\alpha, \beta}$ and $w_T(xy) = w$

-----Finding x, y , $P(u, x)$ and $P(v, y)$ where xy is a BS of uv ; (Lemma 5)-----

```

if  $\mathbf{IS\_Root}(u, p)$  and  $\mathbf{IS\_Root}(v, q)$  # $O(1)$ 
{
     $a, ua \leftarrow \mathbf{FND\_min}(u, p)$  # $O(1)$ 
     $b, vb \leftarrow \mathbf{FND\_min}(v, q)$  # $O(1)$ 
    if  $q'(\overrightarrow{vb})w_T(vb) < p'(\overrightarrow{ua})w_T(ua)$  # $O(1)$ 
        { $x, P(u, x) \leftarrow u, u$ 
          $y, P(v, y) \leftarrow b, vb$ }
    else
        { $x, P(u, x) \leftarrow a, ua$ 
          $y, P(v, y) \leftarrow v, v$ }
}
else
{ $x, P(u, x) \leftarrow \mathbf{Root\_Path}(u, p)$  #  $O(\log n)$  on ave
  $y, P(v, y) \leftarrow \mathbf{Root\_Path}(v, q)$  #  $O(\log n)$  on ave}

```

-----Computing $W_{\alpha, \beta}(T_{xy \setminus uv})$ ----- (Lemma 4)-----

```

 $W'_{\alpha, \beta}(T_{xy \setminus uv}) = \mathbf{Integ}(P(u, x), p_{\alpha, \beta}) + \mathbf{Integ}(P(v, y), q_{\alpha, \beta}) + FN_{\alpha, \beta}(uv)[w - w_T(uv)]$ 
# $O(|P_{T_u^v}(u, x)|) + O(|P_{T_v^u}(v, y)|)$ 
 $W_{\alpha, \beta}(T) = \alpha W_{\times}(T) + \beta W_{+}(T)$  # $O(1)$ 
 $W_{\alpha, \beta}(T_{xy \setminus uv}) = W'_{\alpha, \beta}(T_{xy \setminus uv}) + W_{\alpha, \beta}(T)$  # $O(1)$ 

```

$$\begin{aligned}
& \text{-----} Q_T \leftarrow Q_{T_{xy \setminus uv}} \text{-----updating } Q_T \text{-----} (\text{Lemma 5}) \text{-----} \\
& \quad \text{Up_FN}(P(u, x), f_T(T_v^u), n_T(T_v^u)) \wedge \text{Up_FN}(P(v, y), f_T(T_u^v), n_T(T_u^v)) \\
& \quad \quad \quad \# O(|P_{T_u^v}(u, x)| + |P_{T_v^u}(v, y)|) \\
& W'_{1,0}(T_{xy \setminus uv}) = W'_x(T_{xy \setminus uv}) \\
& \quad = \text{Integ}(P(u, x), p_{1,0}) + \text{Integ}(P(v, y), q_{1,0}) + FN_{1,0}(uv)[w - w_T(uv)] \\
& \quad \quad \quad \# O(|P_{T_u^v}(u, x)| + |P_{T_v^u}(v, y)|) \\
& W'_{0,1}(T_{xy \setminus uv}) = W'_+(T_{xy \setminus uv}) \\
& \quad = \text{Integ}(P(u, x), p_{0,1}) + \text{Integ}(P(v, y), q_{0,1}) + FN_{0,1}(uv)[w - w_T(uv)] \\
& \quad \quad \quad \# O(|P_{T_u^v}(u, x)| + |P_{T_v^u}(v, y)|) \\
& W_x(T_{xy \setminus uv}) = W'_x(T_{xy \setminus uv}) - W_x(T) \quad \# O(1) \\
& W_+(T_{xy \setminus uv}) = W'_+(T_{xy \setminus uv}) - W_+(T) \quad \# O(1) \\
& \text{-----} T \leftarrow T_{xy \setminus uv} \text{-----} \\
& T \leftarrow T + xy - uv \quad \# O(1) \\
& \text{-----}
\end{aligned}$$

Using 1 to 6 above, the average time complexity of every line of Algorithm 2. is bounded by, $\max\{O(|P_{T_u^v}(u, x)|), O(|P_{T_v^u}(v, y)|)\}$. By [25], the average path length of T is $O(\log n)$, which accounts for the worst average time complexity of the algorithm. The best average complexity can be $O(1)$ because $\max\{O(|P_{T_u^v}(u, x)|), O(|P_{T_v^u}(v, y)|)\}$ and the degrees on $P_{T_u^v}(u, x)$ and $P_{T_v^u}(v, y)$ can be $O(1)$. Finally, assume T' is a tree, and we have the preprocessing, $Q_{T'} = \{F_{T'}, N_{T'}, W_x(T'), W_+(T')\}$. If we change the weight of one edge, then $F_{T'}$ and $N_{T'}$ remain unchanged, and by Corollary 4, we can update W_x and W_+ in $O(1)$ -time. This completes the proof. $\square$

The subsequent two propositions can be proved based on Proposition 1. They are indeed a solution to Problems 2 and 3, respectively.

Proposition 2. Let T be an n -vertex positively weighted tree with $E(T) = \{e_i\}_{i=1}^n$. If we get $\{w_i, \alpha_i, \beta_i \in \mathbb{R}^{\geq 0}\}_{i=1}^n$, then we can find an r such that $W_{\alpha_r, \beta_r}(T_{\epsilon_r \setminus e_r}) = \min\{W_{\alpha_i, \beta_i}(T_{\epsilon_i \setminus e_i})\}_{i=1}^n$ and compute $W_{\alpha_r, \beta_r}(T_{\epsilon_r \setminus e_r})$, in an average time of $O(n \log n)$ and the best time of $O(n)$, where ϵ_i with $w_i = w(\epsilon_i)$ is a BS of e_i w.r.t. W_{α_i, β_i} , $0 \leq i \leq n$.

Proposition 3. Let T be an n -vertex positively weighted tree with $E(T) = \{e_i\}_{i=1}^n$. If we get $\{w_i, \alpha_i, \beta_i \in \mathbb{R}\}_{i=1}^n$, then we can find a BS, ϵ_i with $w_i = w(\epsilon_i)$ w.r.t. W_{α_i, β_i} for all e_i and compute $W_{\alpha_i, \beta_i}(T_{\epsilon_i \setminus e_i})$, $0 \leq i \leq n$, in an average time of $O(n \log n)$ and the best time of $O(n)$.

For simplicity, we avoided adding some facts for better performance of the BS algorithm. For example, by Corollary 3, a leaf is not a root of a tree with more than 2 vertices. In the BS algorithm, we either require a root or a neighbor of a root. Thus, if T_u^v or T_v^u are not trivial (single vertex), then the version of T without the leaves can be used. That improves the actual performance of the BS algorithm.

Proposition 4. For a simple tree T , there are exactly $W(T)$ distinct switches. More precisely,

$$|\{T_{\epsilon \setminus e} \mid (\epsilon, e) \in E(T^c \cup T) \times E(T) \wedge T_{\epsilon \setminus e} \text{ is a switch}\}| = W(T),$$

and if $xy \in E(T^c \cup T)$ and $uv \in E(T)$,

$$|\{T_{xy \setminus e} \mid e \in E(T) \wedge T_{xy \setminus e} \text{ is a switch}\}| = d_T(x, y),$$

$$|\{T_{\epsilon \setminus uv} \mid \epsilon \in E(T^c \cup T) \wedge T_{\epsilon \setminus uv} \text{ is a switch}\}| = n_T(T_v^u) \cdot n_T(T_u^v).$$

Proof. Let T be a simple tree. Using Definition 5, $xy \in E(T^c) \cup E(T)$ is a switch of $e \in E(T)$, if and only if $e \in P_T(x, y)$. Otherwise, $T_{xy \setminus e}$ is disconnected, indicating that it is not

a switch. Therefore, there are exactly $d_T(x, y)$ distinct switches regarding $xy \in E(T^c \cup T)$. Thus, the total number of distinct switches is as follows:

$$\begin{aligned} & |\{T_{xy \setminus e} \mid (xy, e) \in E(T^c \cup T) \times E(T) \wedge T_{xy \setminus e} \text{ is a switch}\}| \\ &= \sum_{xy \in E(T^c \cup T)} d_T(x, y) = \sum_{\{x, y\} \subseteq V(T)} d_T(x, y) = W(T). \end{aligned}$$

And $|\{T_{xy \setminus e} \mid e \in E(T) \wedge T_{xy \setminus e} \text{ is a switch}\}| = d_T(x, y)$. In addition, $T_{\epsilon \setminus uv}$ is a tree if and only if ϵ is an edge between T_u^v and T_v^u . Thus, $|\{T_{\epsilon \setminus uv} \mid \epsilon \in E(T^c \cup T) \wedge T_{\epsilon \setminus uv} \text{ is a switch}\}| = n_T(T_v^u) \cdot n_T(T_u^v)$. This completes the proof. $\square$

For $uv \in E(T)$ of a simple tree T with n vertices by definition, $n - 1 \leq n_T(T_v^u) \cdot n_T(T_u^v) \leq \frac{n^2}{4}$. Also, by [26–28], $(n - 1)^2 \leq W(T) \leq \binom{n+1}{3}$.

4. Some Comparisons and Discussion

In this section, we compare our method with some existing methods. For brevity, we mean average time complexity when we talk about complexity. The compared methods are not solving the same problems that we did, but it is possible to convert them with some effort.

The top tree method can find a positively weighted tree's median(s). It is almost equivalent to finding the Root of a tree in our method, disregarding α and β . Thus, for an n -vertex tree, T and $uv \in E(T)$, we somehow can convert the top tree methods [7–9] to find a BS w.r.t. $W_{\alpha=1, \beta=0}$ in $O(\log n)$ -time for most cases. However, if uv connects a Root of T_u^v and a Root of T_v^u , then the top tree method does not work. They also do not compute $W_{\alpha, \beta}$ of the switches regularly, which does not let us solve problem 2 even for their specific case of $\alpha = 1, \beta = 0$. Generalizing the top tree method, we may be able to solve problem 1 for a fixed α, β and compute $W_{\alpha, \beta}$, which is overcomplicated intuitively. But, if we renew the initial values of α, β to some α', β' , then we have to repeat their $O(n)$ -time preprocessing, which is costly. As we have seen, our method in the BS algorithm solves problem 1 in $O(\log n)$ for any $\alpha, \beta \in \mathbb{R}^{\geq 0}$, computes $W'_{\alpha, \beta}$ s, and does not require a fresh preprocessing for the change of α, β . Our method updates the preprocessing, Q_T in $O(1)$ -time after an edge-weight change, whereas using top trees, an edge weight update takes up to $O(\log n)$ -time.

For another comparison, we borrow the solution of the best swap edge of multiple-source routing tree algorithms [15,19–21,29,30] for problem 3. Suppose T is a spanning tree of a positively edge-weighted graph G with m edges and n vertices. Then, in [15,19–21,29,30], researchers choose the BS from $E(G) - E(T)$, whereas we choose from $E(T^c)$. Therefore, by choosing G to be K_n , the mentioned algorithms can solve problem 3. Also, they choose the BS w.r.t. as the routing cost rather than its generalization, $W_{\alpha, \beta}$. Let $S \subseteq V(G)$ be a set of sources and $|S| > 1$. Setting $f_G(v) = 1$ for all $v \in S$ and $f_G(v) = 0$ for all $v \in V(G) \setminus S$, results in

$$W_{1, -1}(G) = RC(G, S).$$

That is, by specifying vertices weights as mentioned and letting $\alpha_i = 1, \beta_i = -1$ for $1 \leq i \leq n$, and $G = K_n$, algorithms of [20,21] will solve problem 3. The mentioned algorithms are expensive specifically for a single failed edge, and after updating the edge weights or inserting a BS, their $O(n^2)$ preprocessing is not valid. They also do not compute $W_{\alpha, \beta}$'s. More importantly, regarding the mentioned conversion, they solve problem 3 in more than $O(n^2 \log^2 n)$ -time and up to $O(n^3)$ depending on the number of sources, where $\alpha_i = 1, \beta_i = -1, 1 \leq i \leq n$ and sources are specified. In the terminology of [20–22], we do not limit the number of sources and, $\alpha_i, \beta_i \in \mathbb{R}^{\geq 0}, 1 \leq i \leq n$, where we attain an average time of $O(n \log n)$ and the best time of $O(n)$ for Problem 3.

As a general fact, note that choosing a brute-force strategy and distance matrix for the positively edge-weighted graph, computing $W_{\alpha, \beta}(T)$ takes $O(n^3)$. Moreover, there

are between $n - 1$ to $\frac{n^2}{4}$ switches regarding an edge. Therefore, using the brute-force strategy, we will need between $O(n^4)$ to $O(n^5)$ time to compute a switch with the minimum $W_{\alpha,\beta}$. While our method executes that in $O(\log n)$, with the same flexibility. This difference is significant. Note that the existing discussed methods are comparable to ours with some constraints, but they have much less flexibility.

Our tools are general and can be applied to similar optimization problems. For instance, see [15], where we use our method for the best swap edge of spanning trees. Another interesting problem can be solving the BS problems regarding ${}^kD(T) = \sum_{\{u,v\} \subseteq V(T)} (d(u,v))^k$ by considering the weight ${}^k\sigma_T(v) = \sum_{x \in V(T)} d_T^k(v,x)$ for the vertices of a tree, T . See Appendix A, where we have some computations related to the derivative and integral of, ${}^2\sigma_T$.

Finding the minimum p -sources routing cost spanning tree is an NP -hard problem for any $p > 1$ and is a polynomial problem for $p = 1$ [29–31].

Author Contributions: Methodology, M.H.K.; Resources, A.-H.E. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the NSF Program on Fairness in AI in collaboration with Amazon under grant IIS-1939368. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Amazon.

Data Availability Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Let T be a tree and $ax \in E(T)$. Then, using the definition (${}^k\sigma_T(v) = \sum_{x \in V(T)} d_T^k(v,x)$, $v \in V(T)$):

$${}^2\sigma_T(a) = \sum_{v \in V(T_a)} d_T^2(a,v) + \sum_{v \in V(T_x)} [d_T^2(x,v) + (w(ax))^2 + 2w(ax)d_T(x,v)],$$

$${}^2\sigma_T(x) = \sum_{v \in V(T_a)} [d_T^2(a,v) + (w(ax))^2 + 2w(ax)d_T(a,v)] + \sum_{v \in V(T_x)} d_T^2(x,v).$$

$$\begin{aligned} {}^2\sigma'_T(ax) &= \frac{{}^2\sigma_T(x) - {}^2\sigma_T(a)}{w_T(ax)} = \frac{(w(ax))^2[n_T(T_x^x) - n_T(T_a^a)] + 2w(ax)[\sigma_{T_a^a}(a) - \sigma_{T_x^x}(x)]}{w_T(ax)} \\ &= w(ax)\sigma'_T(ax) + 2[\sigma_{T_a^a}(a) - \sigma_{T_x^x}(x)] \end{aligned}$$

Also, one can use the same idea of Equation (11) and see that,

$$\begin{aligned} {}^2D'(T_{xy \setminus uv}) &= {}^2D(T_{xy \setminus uv}) - {}^2D(T) \\ &= 2[\sigma_{T_u^v}(x)\sigma_{T_v^u}(y) - \sigma_{T_u^v}(u)\sigma_{T_v^u}(v)] \\ &\quad + n_T(T_u^v) \cdot n_T(T_v^u)[w(xy) - w(uv)] \\ &\quad + n_T(T_v^u)[{}^2\sigma_{T_u^v}(x) - {}^2\sigma_{T_u^v}(u)] + n_T(T_u^v)[{}^2\sigma_{T_v^u}(y) - {}^2\sigma_{T_v^u}(v)] \\ &\quad + 2n_T(T_v^u)[\sigma_{T_u^v}(x) - \sigma_{T_u^v}(u)] + 2n_T(T_u^v)[\sigma_{T_v^u}(y) - \sigma_{T_v^u}(v)]. \end{aligned}$$

We had that, σ'_T is consistent, thus by Theorem 2,

$$\int_u^x \sigma'_{T_u^v} d_{T_u^v} = \sigma_{T_u^v}(x) - \sigma_{T_u^v}(u), \quad \text{and} \quad \int_v^y \sigma'_{T_v^u} d_{T_v^u} = \sigma_{T_v^u}(y) - \sigma_{T_v^u}(v).$$

Therefore,

$$\begin{aligned}
{}^2D'(T_{xy \setminus uv}) = & 2 \left[\left(\int_u^x \sigma'_{T_u^v} d_{T_u^v} + \sigma_{T_u^v}(u) \right) \left(\int_v^y \sigma'_{T_v^u} d_{T_v^u} + \sigma_{T_v^u}(v) \right) - \sigma_{T_u^v}(u) \sigma_{T_v^u}(v) \right] \\
& + n_T(T_v^u) \left[\int_u^x {}^2\sigma'_{T_u^v} d_{T_u^v} \right] + n_T(T_u^v) \left[\int_v^y {}^2\sigma'_{T_v^u} d_{T_v^u} \right] \\
& + 2n_T(T_v^u) \left[\int_u^x \sigma'_{T_u^v} d_{T_u^v} \right] + 2n_T(T_u^v) \left[\int_v^y \sigma'_{T_v^u} d_{T_v^u} \right] \\
& + n_T(T_u^u) \cdot n_T(T_v^v) [w(xy) - w(uv)].
\end{aligned}$$

References

- Alexopoulos, C.; Jacobson, J.A. State space partition algorithms for stochastic systems with applications to minimum spanning trees. *Networks* **2000**, *35*, 118–138.
- Aziz, F.; Gul, H.; Uddin, I.; Gkoutos, G.V. Path-based extensions of local link prediction methods for complex networks. *Sci. Rep.* **2020**, *10*, 19848.
- Dong, Z.; Chen, Y.; Tricco, T.S.; Li, C.; Hu, T. Hunting for vital nodes in complex networks using local information. *Sci. Rep.* **2021**, *11*, 9190.
- Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Red Hook, NY, USA, 4–9 December 2017; pp. 1025–1035.
- Zhou, T. Progresses and challenges in link prediction. *iScience* **2021**, *24*, 103217.
- Kanevsky, A.; Tamassia, R.; Battista, G.D.; Chen, J. On-line maintenance of the four-connected components of a graph. In Proceedings of the 32nd FOCS, San Juan, Puerto Rico, 1–4 October 1991; pp. 793–801.
- Alstrup, S.; Holm, J.; de Lichtenberg, K.; Thorup, M. Maintaining information in fully dynamic trees with top trees. *ACM Trans. Algorithm* **2005**, *1*, 243–264.
- Alstrup, S.; Holm, J.; Thorup, M. Maintaining center and median in dynamic trees. In Proceedings of the 8th Scandinavian Workshop on Algorithm Theory, Berlin, Germany, 3–5 July 2000; pp. 46–56.
- Wang, H.-L. Maintaining centdians in a fully dynamic forest with top trees. *Discrete App. Math.* **2015**, *181*, 310–315.
- Gabow, H.N.; Kaplan, H.; Tarjan, R.E. Unique maximum matching algorithms. *J. Algorithms* **2001**, *40*, 159–183.
- Holm, J.; de Lichtenberg, K.; Thorup, M. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge and biconnectivity. *J. ACM* **2001**, *48*, 723–760.
- Nardelli, E.; Proietti, G.; Widmayer, P. Finding all the best swaps of a minimum diameter spanning tree under transient edge failures. *J. Graph Algorithms Appl.* **2001**, *5*, 39–57.
- Baste, J.; Gözüpek, D.; Paul, C.; Sau, I.; Shalom, M.; Thilikos, D.M. Parameterized complexity of finding a spanning tree with minimum reload cost diameter. *Networks* **2020**, *75*, 259–277.
- Bilò, D.; Colella, F.; Gualà, L.; Leucci, S.; Proietti, G. An Improved Algorithm for Computing All the Best Swap Edges of a Tree Spanner. *Algorithmica* **2020**, *82*, 279–299.
- Fischetti, M.; Lancia, G.; Serafini, P. Exact algorithms for minimum routing cost trees. *Networks* **2002**, *39*, 161–173.
- Gfeller, B.; Santoro, N.; Widmayer, P. A Distributed Algorithm for Finding All Best Swap Edges of a Minimum-Diameter Spanning Tree. *IEEE Trans. Dependable Secur. Comput.* **2011**, *8*, 1–12.
- Kobayashi, M.; Okamoto, Y. Submodularity of minimum-cost spanning tree games. *Networks* **2014**, *63*, 231–238.
- Seth, P. Sensitivity analysis of minimum spanning trees in sub-inverse-Ackermann time. In Proceedings of the 16th International Symposium on Algorithms and Computation, Sanya, China, 19–21 December 2005; pp. 964–973.
- Wu, B.Y.; Lancia, G.; Bafna, V.; Chao, K.-M.; Ravi, R.; Tang, C.Y. A polynomial time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.* **1999**, *29*, 761–777.
- Bilò, D.; Gualà, L.; Proietti, G. Finding Best Swap Edges Minimizing the Routing Cost of a Spanning Tree. *Algorithmica* **2014**, *68*, 337–357.
- Wu, B.Y.; Hsiao, C.-Y.; Chao, K.-M. The swap edges of a multiple-sources routing tree. *Algorithmica* **2008**, *50*, 299–311.
- Khalifeh, M.H.; Esfahanian, A.-H. A Faster Algorithm for Finding Best Swap Edges of multiple-source Routing Tree. *Networks* **2023**, submitted.
- Meyerson, A.; Tagiku, B. Minimizing Average Shortest Path Distances via Shortcut Edge Addition. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques: 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21–23, 2009, Proceedings*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 272–285.
- Wiener, H. Structural determination of the paraffin boiling points. *J. Am. Chem. Soc.* **1947**, *69*, 17–20.
- Shen, Z. The average diameter of general tree structures. *Comput. Math. Appl.* **1998**, *36*, 111–130.
- Dunklemann, P.; Entringer, R.C. Average distance, minimum degree and spanning trees. *J. Graph Theory* **2000**, *33*, 1–13.
- Roger, C.E.; Douglas, E.J.; Snyder, D.A. Distance in graphs. *Czech. Math. J.* **1976**, *26*, 283–296.
- McKay, B.D. The expected eigenvalue distribution of a large regular graph. *Linear Algebra Its Appl.* **1981**, *40*, 203–216.

29. Bilò, D.; Gualà, L.; Proietti, G. A faster computation of all the best swap edges of a shortest paths tree. *Algorithmica* **2015**, *73*, 547–570.
30. Wu, B.Y. A polynomial time approximation scheme for the two-source minimum routing cost spanning trees. *J. Algorithms* **2002**, *44*, 359–378.
31. Wu, B.Y. Approximation algorithms for the optimal p-source communication spanning tree. *Discrete App. Math.* **2004**, *143*, 31–42.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.