An optimal algorithm for the weighted backup 2-center problem on a tree

Hung-Lung Wang
Institute of Information and Decision Sciences
National Taipei University of Business

Abstract. In this paper, we are concerned with the weighted backup 2-center problem on a tree. The backup 2-center problem is a kind of center facility location problem, in which one is asked to deploy two facilities, with a given probability to fail, in a network. Given that the two facilities do not fail simultaneously, the goal is to find two locations, possibly on edges, that minimize the expected value of the maximum distance over all vertices to their closest functioning facility. In the weighted setting, each vertex in the network is associated with a nonnegative weight, and the distance from vertex \( u \) to \( v \) is weighted by the weight of \( u \). With the strategy of prune-and-search, we propose a linear time algorithm, which is asymptotically optimal, to solve the weighted backup 2-center problem on a tree.

Keywords: backup 2-center; weighted center; quasiconvex function; prune-and-search

1 Introduction

Facility location problems are widely investigated in the fields of operations research and theoretical computer science. The \( p \)-center problem is a classic one in this line of investigation. Given a graph \( G \) with positive edge lengths, a supply set \( \Sigma \), and a demand set \( \Delta \), the \( p \)-center problem asks for \( p \) elements \( x_1, x_2, \ldots, x_p \) from \( \Sigma \) such that \( \max_{y \in \Delta} \min_{1 \leq i \leq p} d(x_i, y) \) is minimized, where \( d(x, y) \) denotes the distance from \( x \) to \( y \) in \( G \). Conventionally, \( \Delta, \Sigma \in \{ V, A \} \), where \( V \) and \( A \) are the set of vertices and points of \( G \), respectively. A point of a graph is a location on an edge of the graph, and is identified with the edge it locates on and the distance to an end vertex of the edge. The \( p \)-center problem in general graphs, for arbitrary \( p \), is NP-hard [9], and the best possible approximation ratio is 2, unless NP=\( P \) [11]. When \( p \) is fixed or the underlying network is of specific topology, many efficient algorithms were designed [4,5,7].

There are many generalized formulations of the center problem, like the capacitated center problem [1] and the minmax regret center problem [2,3]. The backup center problem is formulated based on the reliability model [12,13], in which the deployed facilities may sometimes fail, and the demands served by these facilities have to be reassigned to functioning facilities. More precisely,
in the backup \( p \)-center problem, facilities may fail with failure probabilities \( \rho_1, \rho_2, \ldots, \rho_p \). Given that the facilities do not fail simultaneously, the goal is to find \( p \) locations that minimize the expected value of the maximum distance over all vertices to their closest functioning facility. We leave the formal problem definition to Section 2. The backup \( p \)-center problem is NP-hard since it is a generalized formulation of the \( p \)-center problem. For \( p = 2 \), Wang et al. \[15\] proposed a linear time algorithm for the problem on trees. When the edges are of identical length, Hong and Kang \[8\] proposed a linear time algorithm on interval graphs. Recently, Bhattacharya et al. \[6\] consider a weighted formulation of the backup 2-center problem, in which each vertex is associated with a nonnegative weight, and the distance from vertex \( u \) to \( v \) is weighted by the weight of \( u \). They proposed \( O(n) \)-, \( O(n \log n) \)-, \( O(n^2) \)-, and \( O(n^2 \log n) \)-time algorithms on paths, trees, cycles, and unicycles, respectively, where \( n \) is the number of vertices in the corresponding graph.

In this paper, we focus on the weighted backup 2-center problem on a tree and design a linear time algorithm to solve this problem. The algorithm is asymptotically optimal, and therefore improves the current best result on trees, given by Bhattacharya et al. \[6\]. The strategy of our algorithm is prune-and-search, which is widely applied in solving distance-related problems \[10, 14\].

The rest of this paper is organized as follows. In Section 2, we formally define the problem and briefly review the result given by Bhattacharya et al. \[6\]. Based on their observations, a further elaboration on the objective function is given. In Section 3, we design the linear time algorithm, and concluding remarks are given in Section 4.

2 Preliminaries

Let \( T = (V, E) \) be a tree, on which each vertex \( v \) is associated with a nonnegative weight \( w_v \), and each edge is associated with a nonnegative length. A location on an edge is identified as a point, and the set of points of \( T \) is denoted by \( A \). The unique path between two points \( u \) and \( v \) is denoted by \( \pi(u, v) \), and the distance \( d(u, v) \) between two points \( u \) and \( v \) is defined to be the sum of lengths of the edges on \( \pi(u, v) \). The weighted distance from vertex \( u \) to point \( a \) is defined as \( w_u d(u, a) \). The eccentricity of a point \( a \) is defined as

\[
\varepsilon(a) = \max_{u \in V} w_u d(u, a),
\]

and the point with minimum eccentricity is said to be the weighted center of \( T \). Note that the weighted center of a tree is unique. For \( U \subseteq V \), the eccentricity of a vertex \( a \) w.r.t. \( U \) is defined as

\[
\varepsilon(a, U) = \max_{u \in U} w_u d(u, a).
\]

Let \( a_1 \) and \( a_2 \) be two points of \( T \). The partition \( \Pi(a_1, a_2) \) of \( V \) is defined as \( (V_1, V_2) \), where \( V_1 = \{ v \in V : d(v, a_1) \leq d(v, a_2) \} \) and \( V_2 = V \setminus V_1 \). A \emph{weighted 2-center} consists of two points \( c_1 \) and \( c_2 \) minimizing

\[
\varepsilon_2(a_1, a_2) = \max\{\varepsilon(a_1, V_1), \varepsilon(a_2, V_2)\},
\]
where \((V_1, V_2) = \Pi(a_1, a_2)\). We denote a weighted 2-center by \(\{c_1, c_2\}\). Unlike the weighted center of a tree, there may be more than one weighted 2-center. Now we are ready to define the weighted backup 2-center problem.

**Problem 1 (the weighted backup 2-center problem).** Given a tree \(T = (V, E)\) and two real numbers \(\rho_1\) and \(\rho_2\) in \([0, 1)\), the weighted backup 2-center problem asks for a point pair \((b_1, b_2)\) minimizing \(\psi_{\rho_1, \rho_2}: A \times A \rightarrow \mathbb{R}\), where

\[
\psi_{\rho_1, \rho_2}(a_1, a_2) \equiv (1 - \rho_1)(1 - \rho_2) \max\{\varepsilon(a_1, V_1), \varepsilon(a_2, V_2)\} + \rho_2(1 - \rho_1)\varepsilon(a_1) + \rho_1(1 - \rho_2)\varepsilon(a_2),
\]

and \((V_1, V_2) = \Pi(a_1, a_2)\).

To ease the presentation, we assume that \(\rho_1 = \rho_2 = \rho\). With the assumption, minimizing \(\psi_{\rho_1, \rho_2}\) is equivalent to minimizing \(\psi: A \times A \rightarrow \mathbb{R}\), where

\[
\psi(a_1, a_2) \equiv (1 - \rho) \max\{\varepsilon(a_1, V_1), \varepsilon(a_2, V_2)\} + \rho(\varepsilon(a_1) + \varepsilon(a_2)).
\]

We note here that all the proofs in this paper can immediately be extended to the case where failure probabilities are different. Moreover, \(b_1\) and \(b_2\) may be identical, and if \(b_1\) and \(b_2\) are identical, it must be the weighted center, as shown in Proposition 1.

**Proposition 1.** Let \((b_1, b_2)\) be a weighted backup 2-center of tree \(T\). If \(b_1\) and \(b_2\) are identical, then it is the weighted center of \(T\).

**Proof.** Let \(c\) be the weighted center of \(T\). Suppose to the contrary that \(b_1 = b_2\), but \(b_1 \neq c\). Since \(b_1 = b_2\), we have \(\{v \in V: d(v, b_1) \leq d(v, b_2)\} = V\), and therefore

\[
\psi(b_1, b_2) = (1 - \rho) \max\{\varepsilon(b_1, V), \varepsilon(b_2, \emptyset)\} + \rho(\varepsilon(b_1) + \varepsilon(b_2))
\]

\[
= (1 - \rho)\varepsilon(b_1) + \rho(\varepsilon(b_1) + \varepsilon(b_2))
\]

\[
> (1 - \rho)\varepsilon(c) + \rho(\varepsilon(c) + \varepsilon(c))
\]

\[
= \psi(c, c),
\]

which contradicts that \((b_1, b_2)\) is the weighted backup 2-center. \(\square\)

When computing a weighted backup 2-center, any vertex with weight zero can be treated as a point on an edge, and any edge \(uv\) with length zero can be contracted to be a vertex with weight \(\max\{w_u, w_v\}\). With this manipulation, an instance with “nonnegative constraints” on vertex weights and edge lengths can be reduced to one with “positive constraints”, and there is a straightforward correspondence between the solutions. Therefore, in the discussion below, we may focus on the instances with positive vertex weights and edge lengths.
Fig. 1. A tree \( T = (V, E) \). The number beside each vertex and each edge is the weight of the vertex and the length of the edge, respectively. Edges associated with no number are of length one.

2.1 A review on Bhattacharya’s algorithm

Throughout the rest of this paper, we use the tree given in Figure 1 as an illustrative example. Bhattacharya’s algorithm depends on the following observations.

Lemma 1 (See [6]). Let \( \{c_1, c_2\} \) be any 2-center. There is a backup 2-center \( (b_1, b_2) \) such that \( b_1 \) (resp. \( b_2 \)) lies on a path between \( c_1 \) (resp. \( c_2' \)) and \( c \).

Lemma 2 (See [6]). If \( \rho > 0 \), then \( \varepsilon(b_1, V_1) = \varepsilon(b_2, V_2) \) holds for a backup 2-center \( (b_1, b_2) \) on a tree, where \( (V_1, V_2) = \Pi(b_1, b_2) \).

By Lemma 1, we may focus the nontrivial case where \( c_1 < c < c_2 \). The path \( \pi(c_1, c_2) \) is embedded onto the \( x \)-axis with each point \( a \) on \( \pi(c_1, c_2) \) corresponding to point \( d(a, c_1) \) on the \( x \)-axis. For simplicity, \( \pi(c_1, c_2) \) is referred to as both the set of points on this path and the corresponding set of points on the \( x \)-axis. For each vertex \( v \), the cost function \( f_v(x): \pi(c_1, c_2) \mapsto \mathbb{R} \) is defined as

\[
    f_v(x) = w_v d(v, x).
\]

Clearly, \( f_v \) is a V-shape function whose minimum occurs at the point \( a_v^* \) on \( \pi(c_1, c_2) \) closest to \( v \). Let \( f^+_v: \pi(c_1, c_2) \mapsto \mathbb{R} \) and \( f^-_v: \pi(c_1, c_2) \mapsto \mathbb{R} \) be defined as

\[
    f^+_v(x) = \begin{cases} 
    f_v(x), & \text{if } x \geq a_v^* \\
    -\infty, & \text{otherwise,}
    \end{cases}
\]

and

\[
    f^-_v(x) = \begin{cases} 
    f_v(x), & \text{if } x \leq a_v^* \\
    -\infty, & \text{otherwise,}
    \end{cases}
\]

and let the upper envelopes of \( \{f^+_v: v \in V\} \) and \( \{f^-_v: v \in V\} \) be denoted by \( E_L \) and \( E_R \), respectively. An example is given in Figure 2(a). With Lemma 1, we have \( b_1 \in \pi(c_1, c) \) and \( b_2 \in \pi(c, c_2) \). Moreover, the center and 2-center of \( T \) have the following property.

Property 1 (See [6]). For the weighted center \( c \), we have \( \varepsilon(c) = E_L(c) = E_R(c) \). In addition, there is a weighted 2-center \( \{c_1, c_2\} \) satisfying \( \varepsilon(c_1, V_1) = E_L(c_1) \leq \varepsilon(c) \) and \( \varepsilon(c_2, V_2) = E_R(c_2) \leq \varepsilon(c) \), where \( (V_1, V_2) = \Pi(c_1, c_2) \).
In the following, we assume that the weighted 2-center we consider satisfies Property 1. Moreover, it can be derived from Property 1 that for any point pair \((x_1, x_2)\) with \(x_1 \in \pi(c_1, c), x_2 \in \pi(c, c_2)\), and \(E_L(x_1) = E_R(x_2)\), the partition \((V_1, V_2) = \Pi(x_1, x_2)\) satisfies \(\varepsilon(x_1, V_1) = E_L(x_1) = E_R(x_2) = \varepsilon(x_2, V_2)\) since \(\{c_1, c_2\}\) is a weighted 2-center. As a result, Bhattacharya et al. gave the following algorithm to compute a weighted backup 2-center on a tree **T**.

**BU2Center-Tree**(**T**)  
1. \(c \leftarrow\) weighted center of **T**  
2. \(\{c_1, c_2\} \leftarrow\) weighted 2-center of **T**  
3. compute \(E_L\) and \(E_R\)  
4. for each bending point \(x_1\)  
5. \(\text{do } x_2 \leftarrow x^*,\) where \(E_L(x_1) = E_R(x^*)\)  
6. evaluate \(\psi(x_1, x_2)\), and keep the minimum  
7. return \((x_1, x_2)\)

This algorithm runs in \(O(n \log n)\) time. The bottleneck is the computation of \(E_L\) and \(E_R\). Once \(E_L\) and \(E_R\) are computed, the remainder can be done in \(O(n)\) time since there are \(O(n)\) bending points, at which the function \(\psi(x, x^*)\) is not differentiable w.r.t. \(x\). While processing \(x_1\) from left to right, the corresponding \(x_2\) moves monotonically to the left, and therefore a one-pass scan is sufficient to find the optimal solution. Readers can refer to \cite{6} for details. To improve the time complexity, we elaborate some properties of the objective function below.

### 2.2 Properties

As in \cite{6}, the discussion below focuses on the behavior of the objective function on \(\pi(c_1, c_2)\). We observe that the objective function possesses a good property (the
Proposition 2. For any tree $T = (V, E)$, there is a weighted 2-center \{c_1, c_2\} satisfying $\varepsilon(c_1, V_1) = \varepsilon(c_2, V_2)$, where $(V_1, V_2) = \Pi(c_1, c_2)$.

Proof. Let \{c'_1, c'_2\} be a weighted 2-center, and without loss of generality assume that $\varepsilon(c'_1, V'_1) < \varepsilon(c'_2, V'_2)$, where $(V'_1, V'_2) = \Pi(c'_1, c'_2)$. We embed $\pi(c'_1, c'_2)$ onto the $x$-axis as in Section 2.1 and claim that \{a, c'_2\} is a requested weighted 2-center, where

$$a = \min\{x: f^+_E(x) = \varepsilon(c'_2, V'_1), v \in V'_1\}.$$

Notice that \{x: f^+_E(x) = \varepsilon(c'_2, V'_1), v \in V'_1\} is nonempty since $\varepsilon(c'_1, V'_1) < \varepsilon(c'_2, V'_2)$. Let $(U_1, U_2) = \Pi(a, c'_2)$. Clearly, $V'_1 \subseteq U_1$ and $U_2 \subseteq V'_2$. Moreover, $\varepsilon(c'_2, U_2) = \varepsilon(c'_2, V'_1)$.

Suppose to the contrary that \{a, c'_2\} is not the requested weighted 2-center. According to the assumption, either (i) \{a, c'_2\} is not a weighted 2-center, or (ii) $\varepsilon(a, U_1) \neq \varepsilon(a, U_2)$. For (i), there is a vertex $u \in U_1 \setminus V'_1$ satisfying

$$\varepsilon(c'_2, V'_2) = \varepsilon(c'_2, U_2) < \varepsilon(a, U_1) = f_u(a) \leq f_u(c'_2) \leq \varepsilon(c'_2, V'_2),$$

a contradiction. For (ii), it can be derived that $\varepsilon(c'_2, V'_2) = \varepsilon(a, U_1) > \varepsilon(c'_2, U_2) = \varepsilon(c'_2, V'_1)$, which is also a contradiction. \hfill \Box

In the rest of this paper, we assume that \{c_1, c_2\} is a weighted 2-center satisfying $\varepsilon(c_1, V_1) = \varepsilon(c_2, V_2)$, where $(V_1, V_2) = \Pi(c_1, c_2)$. It is noted that once a weighted 2-center is computed, \{c_1, c_2\} can then be computed in linear time, based on the arguments in the proof of Proposition 2.

Next, we elaborate $E_L$ and $E_R$ on $\pi(c_1, c_2)$. As noted in [6], both $E_L$ and $E_R$ are piecewise linear, as summarized in Property 2.

**Property 2.** $E_L$ is strictly increasing and piecewise linear on $\pi(c_1, c_2)$; $E_R$ is strictly decreasing and piecewise linear on $\pi(c_1, c_2)$.

On a path, both $E_L$ and $E_R$ are obviously continuous and convex. We show in Lemma 3 that these properties hold on $\pi(c_1, c_2)$ in a tree.

**Lemma 3.** Let \{c_1, c_2\} be a 2-center of a tree satisfying $\varepsilon(c_1, V_1) = \varepsilon(c_2, V_2)$, where $(V_1, V_2) = \Pi(c_1, c_2)$. The function $E_L$ and $E_R$ are continuous and convex on $\pi(c_1, c_2)$.

Proof. Because of symmetry, we prove the lemma only for $E_L$, and we claim that $E_L$ is continuous. The convexity follows immediately since $E_L$ is the upper envelope of half lines of positive slope. Suppose to the contrary that $E_L$ is not continuous. There is a point $a$ satisfying

$$\lim_{{x \to a^-}} E_L(x) < E_L(a).$$
Let $E_L(a) = f_+^+(a)$. Clearly, at point $a$ we have

$$f_+^+(a) = f_-(a).$$  \hspace{1cm} (1)

Moreover, by the definition of $E_L$, we have $c_1 < a$. If $a = c_2$, then $E_R(a) \geq E_L(a)$, which contradicts $c < c_2$. Therefore, we have

$$f_-(c_2) < f_-(a) < f_-(c_1),$$  \hspace{1cm} (2)

and

$$f_+^+(c_1) < f_+^+(a) < f_+^+(c_2).$$  \hspace{1cm} (3)

Together with Property 2, it can also be derived that

$$E_L(c_1) < E_L(a) < E_L(c_2).$$  \hspace{1cm} (4)

If $v \in V_1$, then

$$E_L(a) = f_-(a) < f_-(c_1) \leq \varepsilon(c_1, V_1) = E_L(c_1) < E_L(a),$$

a contradiction. Otherwise, $\varepsilon(c_2, V_2) = \varepsilon(c_1, V_1) = E_L(c_1) < E_L(a) = f_+^+(a) \leq f_+^+(c_2) \leq \varepsilon(c_2, V_2)$, again a contraction. \hfill \Box

Notice that Lemma 3 does not hold for all 2-centers.

By Lemma 2, the optimal solution occurs at a point pair $(a, a^*)$ satisfying $a \in \pi(c_1, c)$, $a^* \in \pi(c, c_2)$, and $E_L(a) = E_R(a^*)$. Thus, we may focus on a single variable function $\psi_1: \pi(c_1, c) \rightarrow \mathbb{R}$, defined as

$$\psi_1(a) = \psi(a, a^*).$$

To design an efficient algorithm, we expect some good properties on $\psi_1$. Unlike the eccentricity function $\varepsilon$, function $\psi_1$ is not convex on $\pi(c_1, c)$ (see Figure 2(b)). Fortunately, it is quasiconvex. Moreover, for any interval $[c_1, x^*]$ with $c_1 \leq x^* \leq c$, if there is no more than one point at which $\psi_1$ attains the minimum, then $\psi_1$ is strictly quasiconvex on $[c_1, x^*]$ (see Lemma 4).

**Lemma 4 (strict quasiconvexity).** For $a_1 < a_2 < a_3$, the following statements hold.

- $\psi_1(a_3) < \psi_1(a_2)$ implies $\psi_1(a_2) < \psi_1(a_1)$;
- $\psi_1(a_1) < \psi_1(a_2)$ implies $\psi_1(a_2) < \psi_1(a_3)$.

**Proof.** We prove only the statement that $\psi_1(a_3) < \psi_1(a_2)$ implies $\psi_1(a_2) < \psi_1(a_1)$. The other statement can be proved in a similar way. With the assumption that $\psi_1(a_3) < \psi_1(a_2)$, we have

$$(1 - \rho)E_L(a_3) + \rho(E_R(a_3) + E_L(a_2^*)) < (1 - \rho)E_L(a_2) + \rho(E_R(a_2) + E_L(a_2^*)),$$
and therefore

\[ \frac{1 - \rho}{\rho} < \frac{E_R(a_2) - E_R(a_3) + E_L(a_2^*) - E_L(a_3^*)}{E_L(a_3) - E_L(a_2)} \]

\[ = \frac{E_R(a_2) - E_R(a_3)}{E_L(a_3) - E_L(a_2)} + \frac{E_L(a_2^*) - E_L(a_3^*)}{E_L(a_3) - E_L(a_2)} \]

\[ = \frac{(E_R(a_2) - E_R(a_3))/d(a_2, a_3) + (E_L(a_2^*) - E_L(a_3^*)/d(a_2^*, a_3^*)}{E_L(a_2) - E_L(a_1)} \]

We note here that Lemma 4 holds in a symmetric manner for function \( \psi \). This property will be used in designing our algorithm, and its proof is similar to that of Lemma 3.

### 3 A linear time algorithm

The bottleneck on the time complexity of Bhattacharya’s algorithm is the computation of \( E_L \) and \( E_R \). Fortunately, due to the strict quasiconvexity of \( \psi_1 \) and the piecewise linearity of \( E_L \) and \( E_R \), one can apply the strategy of prune-and-search \([10, 11]\) to obtain the optimal solution in linear time. The quasiconvexity of a function \( f \) implies that a local minimum of \( f \) is the global minimum of \( f \), and the idea of the prune-and-search algorithm is to search the local minimum over an interval \( [\lambda_1, \lambda_2] \), which is guaranteed to contain the solution. In the search procedure, \( [\lambda_1, \lambda_2] \) is recursively reduced to a subinterval, and once it is reduced, the size of the instance can also be pruned with a fixed proportion.

In more detail, the following steps are executed in each recursive call:

1. Choose a point \( t \) in \( [\lambda_1, \lambda_2] \) appropriately. Initially, \( [\lambda_1, \lambda_2] = [c_1, c] \).
2. Determine whether \( t < b_1, t = b_1, \) or \( t > b_1 \).
3. Depending on the result of step 2 update \( [\lambda_1, \lambda_2] \), and discard a subset of vertices without affecting the local optimality of \( b_1 \) over the updated interval.

The recursive call is claimed to maintains the following invariant:

**Claim (invariant).** After executing the recursive call, the 6-tuple \((U_1, U_2, U_3, U_4, \lambda_1, \lambda_2)\) satisfies (i) \( b_1 \in [\lambda_1, \lambda_2] \), and (ii) for each \( a \in [\lambda_1, \lambda_2] \), \( E_L(a) = \max_{e \in U_1} f_e^+(a) \), \( E_R(a) = \max_{e \in U_2} f_e^-(a) \), \( E_L(a^*) = \max_{e \in U_3} f_e^+(a^*) \), and \( E_R(a^*) = \max_{e \in U_4} f_e^-(a^*) \).
We note here that to guarantee the efficiency of the algorithm, the steps above have to be repeated with $c_1$ and $b_1$ being replaced with $c_2$ and $b_2$, respectively, in each recursive call. The reason will be clear after the elaboration below. The details of steps 1 and 3 are given in Section 3.1, that of step 2 is given in Section 3.2, and the analysis of the algorithm is given in Section 3.3.

### 3.1 Guaranteeing the discarded proportion of vertices

The proportion of vertices discarded at step 3 depends essentially on how the point $t$ at step 1 is chosen. According to the piecewise linearity of $E_L$ and $E_R$, the discarded proportion can be guaranteed based on the following simple property.

**Property 3.** Consider two linear functions $f_1(x) = a_1x + b_1$ and $f_2(x) = a_2x + b_2$ with $a_1 > a_2$. We have $f_1(x) \leq f_2(x)$ if and only if $x \leq \frac{b_2 - b_1}{a_1 - a_2}$.

As the idea in [10, 14], for $i \in \{1, 2\}$, we arbitrarily partition $U_i$ into $\lfloor |U_i|/2 \rfloor$ pairs of vertices, and a single one if $|U_i|$ is odd, and let this partition be denoted by $\Pi(U_i)$. Moreover, let $t^+_uv = \nu(f^+_u, f^+_v)$ and $t^-uv = \nu(f^-u, f^-v)$, where $\nu(f_1, f_2)$ denotes the point at which $f_1$ and $f_2$ intersect. If $t_m$, the median of $\left\{ t^+_uv : \{u, v\} \in \Pi(U_1) \right\} \cup \left\{ t^-uv : \{u, v\} \in \Pi(U_2) \right\}$, is chosen at step 1, then after step 2, $\left\lfloor \frac{\lfloor |U_1|/2 \rfloor + |U_2|/2}2 \right\rfloor$ vertices can be discarded without affecting the optimality of $b_1$ in either $[\lambda_1, \min\{t_m, \lambda_2\}]$ or $[\max\{t_m, \lambda_1\}, \lambda_2]$. The procedures are given as **PairPartitionMedian** and **DiscardVertices**, respectively. An illustration is given in Figure 3.

---

**Fig. 3.** (a) The instance before the execution of **ReduceInstance1**(V, V, V, V, c_1, c). In the execution, every two vertices in $U_1$, from left to right, is paired in the partition $\Pi(U_1), \{t^+_uv : \{u, v\} \in \Pi(U_1)\} \cup \{t^-uv : \{u, v\} \in \Pi(U_2)\} = \{3, -2, 49/4, 29/4, 5/2\} \cup \{-3, 2/3, 39/4, 25/4, 21/2\}$, and $t$ is chosen as $37/8$. (b) The instance after the execution of **ReduceInstance1**(V, V, V, V, c_1, c). Hollow circles denote the discarded elements from $U_1 \cup U_2$. 

(a) (b)
**PAIRPARTITIONMEDIAN**(\(U_1, U_2\))

1. \(X_1 \leftarrow \Pi(U_1)\)
2. \(X_2 \leftarrow \Pi(U_2)\)
3. \(t_m \leftarrow \text{the median of } \{t_{uv}^+ : \{u, v\} \in X_1\} \cup \{t_{uv}^- : \{u, v\} \in X_2\}\)
4. **return** \((t_m, X_1, X_2)\)

**DISCARDVERTICES**(\(X_1, X_2, t, \lambda_1, \lambda_2\))

1. \(U_1 \leftarrow \emptyset\)
2. \(U_2 \leftarrow \emptyset\)
3. if \(t \leq \lambda_1\) ⚫ \([\lambda_1, \lambda_2]\) is updated, so either \(t \leq \lambda_1\) or \(t \geq \lambda_2\)
   4. then for each element \(x\) in \(X_1\)
      5. do if \(x = \{u, v\}\) and \(\nu(f_u^+, f_v^+) \leq t\)
         6. then \(U_1 \leftarrow U_1 \cup \{u: f_u^-(t + \varepsilon) > f_v^-(t + \varepsilon)\}\)
         7. else \(U_1 \leftarrow U_1 \cup x\)
   8. for each element \(x\) in \(X_2\)
      9. do if \(x = \{u, v\}\) and \(\nu(f_u^-, f_v^-) \leq t\)
         10. then \(U_2 \leftarrow U_2 \cup \{u: f_u^-(t + \varepsilon) > f_v^-(t + \varepsilon)\}\)
         11. else \(U_2 \leftarrow U_2 \cup x\)
   12. else for each element \(x\) in \(X_1\)
      13. do if \(x = \{u, v\}\) and \(\nu(f_u^+, f_v^-) \geq t\)
         14. then \(U_1 \leftarrow U_1 \cup \{u: f_u^+(t - \varepsilon) > f_v^+(t - \varepsilon)\}\)
         15. else \(U_1 \leftarrow U_1 \cup x\)
   16. for each element \(x\) in \(X_2\)
      17. do if \(x = \{u, v\}\) and \(\nu(f_u^-, f_v^-) \geq t\)
         18. then \(U_2 \leftarrow U_2 \cup \{u: f_u^-(t - \varepsilon) > f_v^-(t - \varepsilon)\}\)
         19. else \(U_2 \leftarrow U_2 \cup x\)
   20. **return** \((U_1, U_2)\)

### 3.2 Evaluating \(\psi_1\)

Step 2 can be done via evaluating \(\psi_1(t_m)\) and \(\psi_1(t_m - \varepsilon)\). Due to the quasiconvexity of \(\psi_1\) (Lemma 4), we have that

- if \(\psi_1(t_m - \varepsilon) > \psi_1(t_m)\), then \(b_1 > t_m\);
- if \(\psi_1(t_m - \varepsilon) \leq \psi_1(t_m)\), then \(\exists b_1 \leq t_m\).

Recall that \(\psi_1(a) = (1 - \rho)E_L(a) + \rho(E_R(a) + E_L(a^*))\), where \(E_R(a^*) = E_L(a)\). For a point \(a\) on \(\pi(c_1, c)\), the evaluation of \(\psi_1(a)\) can be done via computing \(E_L(a), E_R(a),\) and \(E_L(a^*)\). According to the claim of invariant, for \(a \in [\lambda_1, \lambda_2]\), we have

\[
E_L(a) = \max\{f_u^+(a) : v \in U_1\} \quad \text{and} \quad E_R(a) = \max\{f_v^-(a) : v \in U_2\}.
\]

Clearly, \(E_L(a)\) and \(E_R(a)\) can be computed in time linear to \(|U_1|\) and \(|U_2|\), respectively. If \(a^*\) is given, then \(E_L(a^*)\) can also be computed in \(O(|U_4|)\) time.

It remains to show how \(a^*\) is determined.

10
Given a value $\xi$, since $E_R(a) = \max\{f^-(v): v \in U_3\}$, the point $a$ on $\pi(c, c_2)$ satisfies $E_R(a) = \xi$ if and only if

$$a = \max\{x: f^-_v(x) = \xi, v \in U_3\}.$$ 

Therefore, for $a \in [\lambda_1, \lambda_2]$, determining $a^*$ can be done in $O(|U_3|)$ time, and $E_L(a^*)$ can then be computed in $O(|U_4|)$ time. Formally, the procedure of evaluating $\psi_2$ at point $a$ in $[\lambda_1, \lambda_2]$ is given as Evaluate1($a, U_1, U_2, U_3, U_4$).

**Evaluate1($a, U_1, U_2, U_3, U_4$)**

1. $\xi_1 \leftarrow \max\{f^+_v(a): v \in U_1\}$
2. $\xi_2 \leftarrow \max\{f^-_v(a): v \in U_2\}$
3. $a^* \leftarrow \max\{x: f^-_v(x) = \xi_1, v \in U_4\}$
4. $\xi_3 \leftarrow \max\{f^+_v(a^*): v \in U_3\}$
5. **return** $(1 - \rho)\xi_1 + \rho(\xi_2 + \xi_3)$

The evaluation of $\psi_2$ at a given point $a$ can be done symmetrically, and the details are omitted.

### 3.3 The analysis of the algorithm

With the procedures given in Sections 3.1 and 3.2, we may implement the idea given in the beginning of Section 3, which recursively reduces the size of the problem instance. The procedure is given as ReduceInstance1.

**ReduceInstance1($U_1, U_2, U_3, U_4, \lambda_1, \lambda_2$)**

1. $(t, X_1, X_2) \leftarrow \text{PairPartitionMedian}(U_1, U_2)$
2. $\xi_1 \leftarrow \text{Evaluate1}(t - \epsilon, U_1, U_2, U_3, U_4)$
3. $\xi_2 \leftarrow \text{Evaluate1}(t, U_1, U_2, U_3, U_4)$
4. if $\xi_1 < \xi_2$
5. **then** $\lambda_2 \leftarrow \min\{t, \lambda_2\}$
6. **else** $\lambda_1 \leftarrow \max\{t, \lambda_1\}$
7. $(U_1, U_2) \leftarrow \text{DiscardVertices}(X_1, X_2, t, \lambda_1, \lambda_2)$
8. **return** $(U_1, U_2, \lambda_1, \lambda_2)$

An example is given in Figure [3]. Since $t$ is chosen as the median of $\{t^+_uv: \{u,v\} \in \Pi(U_1)\} \cup \{t^-uv: \{u,v\} \in \Pi(U_2)\}$ (line 1), it can be derived that a fixed proportion of vertices in $U_1 \cup U_2$ are discarded after the execution of DiscardVertices (line 7). Moreover, together with Property 3, it can be easily derived that for $a \in [\lambda_1, \lambda_2]$, $\psi_1(a)$ remains unchanged. We summarize these properties in Lemma 5.

#### Lemma 5.

After the execution of ReduceInstance1($U_1, U_2, U_3, U_4, \lambda_1, \lambda_2$), at least $\left\lfloor \frac{n_1/2 + n_2/2}{2} \right\rfloor$ vertices of $U_1 \cup U_2$ are discarded, where $n_1 = |U_1|$ and $n_2 = |U_2|$. Moreover, the procedure maintains the invariant $(U_1, U_2, U_3, U_4, \lambda_1, \lambda_2)$, in which $b_1 \in [\lambda_1, \lambda_2]$, and for $a \in [\lambda_1, \lambda_2]$, $E_L(a) = \max\{f^+_v(a): v \in U_1\}$, $E_R(a) = \max\{f^-_v(a): v \in U_2\}$, $E_L(a^*) = \max\{f^+_v(a^*): v \in U_3\}$, and $E_R(a^*) = \max\{f^-_v(a^*): v \in U_4\}$.
Notice that after the execution of REDUCEINSTANCE1(U₁, U₂, U₃, U₄, λ₁, λ₂), only a proportion of U₁ ∪ U₂ is discarded. To ensure that a fixed proportion of the instance is pruned, in a symmetric manner, one can apply a similar procedure to discard a proportion of U₃ ∪ U₄, as noted in the beginning of Section [3]. We name the corresponding procedure as REDUCEINSTANCE2. With REDUCEINSTANCE1 and REDUCEINSTANCE2, one may recursively reduces the size of the instance until the instance is small enough. For small instance with both |U₁ ∪ U₂| ≤ 3 and |U₃ ∪ U₄| ≤ 3, we compute the solution by evaluating ψ₁ at all bending points in [λ₁, λ₂] since ψ₁ is piecewise linear. We denote this procedure by SMALLINSTANCE.

The integration is given as PRUNEANDSEARCH, and the procedure for computing a weighted backup 2-center is given as BACKUP2CENTER. The correctness and time complexity are analyzed in Theorem [1].

PRUNEANDSEARCH(U₁, U₂, U₃, U₄, λ₁, λ₂, λ₃, λ₄)
1 if |U₁ ∪ U₂| ≤ 3 and |U₃ ∪ U₄| ≤ 3 then return SMALLINSTANCE(U₁, U₂, U₃, U₄, λ₁, λ₂, λ₃, λ₄)
3 else (U’₁, U’₂, λ’₁, λ’₂) ← REDUCEINSTANCE1(U₁, U₂, U₃, U₄, λ₁, λ₂)
4 (U’₃, U’₄, λ’₃, λ’₄) ← REDUCEINSTANCE2(U₁, U₂, U₃, U₄, λ₃, λ₄)
5 return PRUNEANDSEARCH(U’₁, U’₂, U’₃, U’₄, λ’₁, λ’₂, λ’₃, λ’₄)

BACKUP2CENTER(T = (V, E), ρ)
1 c ← weighted center of T
2 (c₁, c₂) ← weighted 2-center of T
3 return PRUNEANDSEARCH(V, V, V, c₁, c, c₂)

Theorem 1. The weighted backup 2-center problem on a tree can be solved in linear time by BACKUP2CENTER.

Proof. Let T = (V, E) with |V| = n. By Lemma [1] b₁ ∈ [c₁, c] and b₂ ∈ [c, c₂], and thus λ₁, λ₂, λ₃, and λ₄ are initialized accordingly as in BACKUP2CENTER. Besides, by definition, we have E_L(a) = max{f⁺_v(a): v ∈ V} and E_R(a) = max{f⁻_v(a): v ∈ V}. With Lemma [5] the initialization of U₁ = V and U₂ = V guarantees ψ₁(a), for a ∈ [λ₁, λ₂], is computed correctly. Similar arguments hold for the initialization of U₃ = V and U₄ = V.

For the time complexity, both the weighted center and the weighted 2-center can be computed in O(n) time [4][10]. For PRUNEANDSEARCH, let n₁ = |U₁|, n₂ = |U₂|, n₃ = |U₃|, and n₄ = |U₄|. Lines [3][5] of PRUNEANDSEARCH are executed if either |U₁ ∪ U₂| ≥ 4 or |U₃ ∪ U₄| ≥ 4. Together with Lemma [5] it can be derived that

\[
\left\lfloor \frac{|n₁/2| + |n₂/2|}{2} \right\rfloor + \left\lfloor \frac{|n₃/2| + |n₄/2|}{2} \right\rfloor \geq \frac{1}{8}(n₁ + n₂ + n₃ + n₄), \quad (5)
\]

where the equality holds when n₁ = 3, n₂ = 3, n₃ = 1, and n₄ = 1. As a result, let the execution time of PRUNEANDSEARCH(V, V, V, V, c₁, c, c₂) be T(N), where
\[ N = |V| + |V| + |V| + |V| = 4n. \] It follows from (5) that
\[ T(N) \leq T\left(\frac{7N}{8}\right) + O(N). \]

Therefore, \( T(N) = O(N) = O(n). \)

\section{Concluding remarks}

In this paper, we propose a linear time algorithm to solve the weighted backup 2-center problem on a tree, which is asymptotically optimal. Based on the observations given by Bhattacharya et al. \cite{6}, “good properties” of the objective function are further derived. With these properties, the strategy of prune-and-search can be applied to solve this problem. For future research, the hardness of the backup \( p \)-center problem on trees is still unknown, even for the unweighted case. It worth investigation on this direction.

\section*{Acknowledgements}

The author would like to thank Professor Kun-Mao Chao, Ming-Wei Shao, and Jhih-Heng Huang for helpful discussions. Hung-Lung Wang was supported in part by MOST grant 103-2221-E-141-004, from the Ministry of Science and Technology, Taiwan.

\section*{References}

1. M. Albareda-Sambola, J. A. Díaz, and E. Fernández, Lagrangean duals and exact solution to the capacitated \( p \)-center problem, Eur. J. Oper. Res. 201 (2010) 71–81.
2. I. Averbakh and O. Berman, Minimax regret \( p \)-center location on a network with demand uncertainty, Location Science 5 (1997) 247–254.
3. I. Averbakh and O. Berman, Algorithms for the robust 1-center problem on a tree, Eur. J. Oper. Res. 123 (2000) 292–302.
4. B. Ben-Moshe, B. Bhattacharya, Q. Shi, An optimal algorithm for the continuous/discrete weighted 2-center problem in trees, Proc. 7th Latin American Symposium Theoretical Informatics (LNCS 3887), Valdivia, Chile, 2006, pp. 166–177.
5. B. Ben-Moshe, B. Bhattacharya, Q. Shi, and A. Tamir, Efficient algorithms for center problems in cactus networks, Theor. Compt. Sci. 378 (2007) 237–252.
6. B. Bhattacharya, M. De, T. Kameda, S. Roy, V. Sokol, Z. Song, Back-up 2-center on a path/tree/cycle/unicycle, Proc. 20th International Conference on Computing and Combinatorics (LNCS 8591), Atlanta, GA, USA, 2014, pp. 417–428.
7. G. N. Frederickson, Parametric search and locating supply centers in trees, Proc. 2nd Workshop on Algorithms and Data Structures (LNCS 519), Ottawa, Canada, 1991, pp. 299–319.
8. Y. Hong and L. Kang, Backup 2-center on interval graphs, Theor. Comput. Sci. 445 (2012) 25–35.
9. O. Kariv and S.L. Hakimi, An algorithmic approach to network location problems, Part I. The $p$-centers, SIAM J. Appl. Math. 37 (1979) 441–461.
10. N. Megiddo, Linear-time algorithms for linear programming in $R^3$ and related problems, SIAM J. Comput. 12 (1983) 759–776.
11. J. Plesník, A heuristic for the $p$-center problem in graphs, Discrete Appl. Math. 17 (1987) 263–268.
12. L. V. Snyder, Facility location under uncertainty: A review, IIE Trans. 38 (2006) 537–554.
13. L. V. Snyder and M. S. Daskin, Reliability models for facility location: The expected failure cost case, Transport. Sci. 39 (2005) 400–416.
14. A. Tamir, Sorting weighted distances with applications to objective function evaluations in single facility location problems, Oper. Res. Lett. 32 (2004) 249–257.
15. H.-L. Wang, B. Y. Wu, and K.-M. Chao, The backup 2-center and backup 2-median problems on trees, Networks 53 (2009) 39–49.