A Linear Time Algorithm for Absolute Optima under Monotonous Gain Functions on Trees

J. Spoerhase and H.-C. Wirth

October 4, 2007

We examine problems of placing facilities in a tree graph to serve customers. The decision of placement is driven by an election process amongst the users, where the user preference is modeled by distances in the tree. Relaxed user preferences introduce a tolerance of users against small differences in distances. Monotonous gain functions are a natural generalization of well known competitive and voting location problems such as Simpson, security, Stackelberg and Nash.

In this technical report we present an algorithm which computes an absolute solution, i.e., a point with minimum score for any monotonous gain function. The running time is linear in the size of the input tree and hence optimal.

1 Introduction and Preliminaries

For an introduction into competitive and voting problems on graphs and a motivation of the monotonous gain functions we refer the reader of this technical report to the outline given in [SW07] and the references cited therein, in particular [CM03].

Consider a tree $T = (V, E)$ with a positive edge weight function $d\colon E \to \mathbb{R}^+$ denoting the length of each edge. A *point* x on an edge $e = (u, v)$ is determined by the distance $0 \leq d(u, x) \leq d(u, v)$ and satisfies the invariant $d(u, x) + d(x, v) = d(u, v)$. For a tree T (or an edge e) we use T (or e) to denote both the tree (or the edge) and the set of all of its points, as the meaning will be clear from the context. This induces a distance function $d\colon T \times T \to \mathbb{R}_0^+$ on the set of all points.

The input instance of the problem under investigation is given as a tree $T_0 = (V, E)$ with edge lengths $d\colon E \to \mathbb{R}^+$. Non-negative node weights $w\colon V \to \mathbb{R}_0^+$ specify the demand of individual user nodes. A non-negative number $\alpha \in \mathbb{R}_0^+$ is used as a parameter to describe the users' tolerance against small differences in distances as follows:

Definition 1.1 (Relaxed user preference) A user u *prefers* node x over node y, denoted by $x \prec_u y$, if

$$d(u,x) < d(u,y) - \alpha\,. \tag{1}$$

The user u *is undecided*, $x \sim_u y$, if $|d(u,x) - d(u,y)| \leq \alpha$.

We use the following notation: The set of users preferring x over y is denoted by $U(x \prec y) := \{\, u \in V \mid x \prec_u y \,\}$, and its weight by $w(x \prec y) := w(U(x \prec y))$.

A *gain function* $\Phi\colon T \times T \to \mathbb{R}$ maps a point pair (y,x) to the value $\Phi(y \prec x)$ which measures in some sense the influence of a follower point y after leader point x has already been placed into the graph. Given a gain function, the notions *absolute score* and *absolute solution* are defined as follows:

Definition 1.2 (Absolute Φ-score and Φ-solution) For any gain function Φ, the *absolute* Φ*-score* of a leader point x is defined as

$$\Phi(x) := \max_{\text{point } y \in T} \Phi(y \prec x)\,.$$

Any point y with $\Phi(y \prec x) = \Phi(x)$ is called a *witness of* x. The *absolute* Φ*-score* of a graph is defined as $\Phi^* := \min_{\text{point } x \in T} \Phi(x)$. An *absolute* Φ*-solution* of a graph is a point x with $\Phi(x) = \Phi^*$.

Clearly this definition is too weak to derive a general algorithm for Φ-score and solution beyond the limits of a trivial enumeration. A minimum requirement is that a gain function reflects the user preference induced by the graph. In particular, if we start with two nodes x, y and move them in such a way that weight $w(y \prec x)$ of the influence area of the follower y increases while the weight $w(x \prec y)$ of the influence area of the leader x decreases, one would expect that this does not decrease the value $\Phi(y \prec x)$. Moreover we desire that Φ can be quickly evaluated once the weights of the influence areas are given. We call gain functions with this property to be *monotonous*:

Definition 1.3 (Monotonous gain function) A gain function $\Phi(y \prec x)$ is called *monotonous*, if there is a function $\varphi\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that

1. $\Phi(y \prec x) = \varphi(w(y \prec x), w(x \prec y))$ for all points $x, y \in T$
2. φ is monotonously increasing in the first parameter and monotonously decreasing in the second parameter
3. φ can be evaluated in constant time

Notice that the last requirement can be also achieved after a preprocessing step which itself may need a super-constant time. Typical monotonous gain functions include the *Simpson score* $\Phi(y \prec x) := w(y \prec x)$, the *security score* $\Phi(y \prec x) := w(y \prec x) - w(x \prec y)$, and the *Stackelberg score* $\Phi(y \prec x) := w(y \prec x) + \frac{1}{2} w(y \sim x)$. We will argue later in Section 3 how to extend this definition and the results obtained to more general models.

We make use of the following standard notation: By $T_u(v)$ we denote the subtree below node v when the tree is rooted at u, and by $w_u(v)$ its weight. This notation is easily

extended to the case where u or v are points, namely by temporarily adding a new node at the position of the point. The *α-ball around point x* is denoted by $S_\alpha(x) := \{\, y \in T \mid d(x,y) \leq \alpha \,\}$.

We briefly report some results on working with monotonous gain functions derived in [SW07]:

Definition 1.4 (α-neighborhood) Let x be a point. Any point y where $d(x,y)$ is infinitesimally greater than α is called an *α-neighbor* of x. The set of all α-neighbors of x is denoted by $N(x,\alpha)$.

As discussed formally in [SW07], this infinitesimality notion is merely a technical detail which reflects the strict inequality in (1). In fact one can determine in advance discrete representants as α-neighbors.

Theorem 1.5 (Witness) *For each point x there is a witness $y \in N(x,\alpha) \cup \{x\}$.* □

Theorem 1.6 (Characterization of leader and follower party) *Let x be a leader point and y be an α-neighbor of x. Then the leader party is $U(y \prec x) = T_x(y)$ and the follower party is $U(x \prec y) = T_y(x)$.* □

As a consequence, the weight of the leader and follower party is always the weight of a subtree hanging from a node. The weights of all those possible subtrees can be precomputed by two depth first search traversals in linear time. Hence we can afterwards evaluate the monotonous gain function in constant time by the equation $\Phi(y \prec x) = \varphi(w_x(y), w_y(x))$ if y is an α-neighbor of x. To simplify the notation we define $\varphi(y,x) := \varphi(w_x(y)), w_y(x))$ for any point pair x, y.

We make further use of a technical *guide rule* outlined in [SW07] which essentially states that given a point x and a witness y the search for a Φ-solution can be restricted to the subtree below x in the direction of y.

Lemma 1.7 (Guide rule) *Let T be a tree, $x \in T$ be a point in the tree, and $y \in T$ be a witness of x where $y \notin S_\alpha(x)$. Then $\Phi(x') \geq \Phi(x)$ for all points $x' \in T_y(x)$. If on the other hand x is a witness of itself, then $\Phi(x) = \Phi^*$.*

Proof. Let $x' \in T_y(x)$ be an arbitrary point. Then $\Phi(x') \geq \Phi(y \prec x') \geq \Phi(y \prec x) = \Phi(x)$ by the monotonicity of Φ. On the second claim observe that if x is a witness of itself then all nodes are undecided, thus $\Phi(x) = \Phi(x \prec x) = \varphi(0,0)$. Moreover, $\varphi(0,0) = \Phi(x' \prec x') \leq \Phi(x')$ for all other points x', hence $\Phi(x)$ is optimal. □

Related Work and Contribution of This Report

In [SW07] we have provided an algorithm with running time $O(n \log n)$ which computes an arbitrary absolute Φ-solution of a tree. That algorithm works for all monotonous gain functions. (Notice that the definition of monotonous gain function introduced in [SW07] has been slightly tightened in the current report to cover only interesting functions.)

In the current report we improve this result and solve the same problem optimally by providing a linear time algorithm.

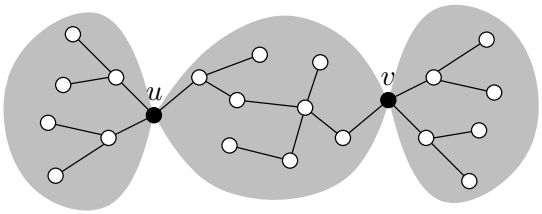

Figure 1: Example of a terminal tree with terminals u, v.

2 Linear Time Algorithm

In this section we develop an algorithm which computes the absolute Φ-score of a tree in linear time for any monotonous gain function Φ. The rough idea of the algorithm is to maintain a *leader tree* which is always a subtree of the input tree and is guaranteed to contain a Φ-solution, i.e., an optimal placement for the leader. The leader tree is initialized with the input tree and iteratively decreased during the execution of the algorithm. When the number of remaining nodes has reached $O(1)$, the iteration stops and a solution can be found and output in constant time.

The algorithm ensures the invariant that for points in the current leader tree the Φ-score with respect to the original tree is always a lower bound of the current Φ-score, and that for optimal points the scores are identical. Therefore nodes which are identified as irrelevant for the leader cannot simply be discarded from the tree as they might still be relevant for the follower: if such a node serves currently as a witness for some potential leader point x its removal could decrease the Φ-score of x which is undesired. To account for this problem those nodes are organized in two *follower trees*. Even the follower trees are decreased in size during the execution of the algorithm. However, when the algorithm discards nodes from one of the follower trees it takes care that essential witnesses are only removed when there remains a suitable substitution such that the above mentioned invariant is maintained.

The linear running time is achieved since first each iteration takes a time linear in the size of the current tree and second that size decreases bounded from above by a falling geometric sequence during all iterations.

The actual data structure employed by the algorithm is called a *terminal tree* and defined as follows (see Figure 1 for an illustration):

Definition 2.1 (Terminal tree) A *terminal tree* is a tree $T = (V, E)$ with two distinguished nodes $u, v \in V$, called the *terminals*. The maximal subtree of T containing both terminals as leaves is called the *leader tree*. The trees $F_u := T_u(v)$ and $F_v := T_v(u)$ are called *follower trees*.

The algorithm maintains a terminal tree which is initialized to the input tree T_0 where two arbitrary leaves are chosen as initial terminals. Later modifications are designed in a way that the current leader tree always contains at least one optimal point. This

invariant is formalized by the notion of Φ-*equivalence*. To this end we use subscript notation Φ_G to denote the Φ-score in a graph G.

Definition 2.2 (Φ-equivalent terminal tree) Let T be a terminal tree with leader tree L. The tree T is called Φ-*equivalent to* T_0 if $\Phi_T(x) \geq \Phi_{T_0}(x)$ for all points $x \in L$ and $\min_{x \in L} \Phi_T(x) = \Phi^*_{T_0}$.

Let T be a terminal tree with leader tree L which is Φ-equivalent to input tree T_0. The leader tree is a subtree of the input tree, hence each point in L can be considered as a point in T as well. A point $x^* \in L$ with $\Phi_T(x^*) = \min_{x \in L} \Phi_{T_0}(x)$ is called an L-*optimum*. Each L-optimum in the current tree is also an optimal point in the original tree T_0 with respect to Φ_{T_0}. (Notice that the converse does not hold in general.) To find an optimum placement for the leader (with respect to the original tree) it thus suffices to restrict the view to the current leader tree L.

The main purpose of the follower tree is to collect nodes removed from the leader tree which may be needed to furnish certain nodes of the leader tree with a suitable witness in order to prevent that their Φ-score decreases. To this end it is not required that the follower trees are actually subtrees of T_0 and in fact their structure can be completely different from that of the corresponding part of the input tree.

Let $x \in L$ be a point in the leader tree with a witness y contained in the follower tree F. Then $\Phi(y \prec x) = \varphi(y', x)$ where y' is a node where the tree weight $w_x(y')$ is maximum among all nodes in F with distance greater than α to x. Hence Φ-equivalence is tantamount to the property that the weight of the heaviest subtree with distance at least α to a given point $x \in L$ is the same (or greater as long as L retains the optimum property in the definition of Φ-equivalence) as in the original tree T. This fact allows us to sparsen the follower trees iteratively without violating the Φ-equivalence.

2.1 Sparsen the Follower Tree

In the sequel we describe a linear time operation which halves the size of a follower tree but does not affect the Φ-equivalence of the current terminal tree T. To this end let T be the current terminal tree, L be its leader tree, u be one of its terminals, and $\delta > 0$ be a distance. As previously stated F_u denotes the follower tree incident with terminal u. We define subsets $F^+, F^- \subseteq F_u$ by

$$F^+ := \{\, y \in F_u \mid d(u,y) \geq \delta \,\} \qquad \text{and} \qquad F^- := \{\, y \in F_u \mid d(u,y) \leq \delta \,\}\,.$$

We will later choose δ as the median of the distances of all nodes of the follower tree to its terminal and this way divide the set of follower nodes in two almost equal sized parts. (It is guaranteed in particular that both F^+ and F^- are nonempty.)

The main idea is to reduce the size of the follower tree by essentially discarding either the nodes in F^+ or in F^- from it. As outlined before simply removing the nodes can decrease the Φ-score of leader nodes thus invalidating the Φ-equivalence property which is undesired. To this end we employ two subroutines DiscardNear (see Figure 4) and DiscardFar (see Figure 5) which discard the nodes in F^- or F^+, respectively, while

taking further care that the Φ-scores of leader points do not decrease which is a main ingredient to maintain the Φ-equivalence property.

We first determine a point $h \in F^+$ with $d(u,h) = \delta$ such that $w_u(h)$ is maximal. For later reference we denote the subset of leader points in the α-ball around h by $S := S_\alpha(h) \cap L$. For any point set $X \subseteq L$ denote by ∂X its boundary with respect to the point space (L,d).

It is an easy observation that leader points in S do not have α-neighbors in F^- and hence in particular no witnesses in F^-. Thus if S contains an L-optimum then DiscardNear maintains the Φ-equivalence of T. Otherwise there is an L-optimum outside of S such that the follower has no incentive to choose a location in F^+. Hence DiscardFar is a legal operation. A straightforward test to distinguish both cases would work as follows. Consider each $x \in \partial S$ as leader candidate and determine a witness y for x. If $y \in T_h(x)$ for some x then the second case applies due to the guide rule. Otherwise if no $x \in \partial S$ has a witness below x the set S must contain an L-optimum and we may execute DiscardNear. Unfortunately this approach is too expensive since ∂S might contain $\Theta(n)$ nodes and thus the running time would be no longer linear.

In order to overcome these difficulties we introduce a set $L^+ \subseteq S$ with the property that each element outside of L^+ has no incentive to choose a witness in F^+. Moreover we are going to show that it is possible to identify a single candidate in the boundary ∂L^+ such that computing its witness is already sufficient to reveal the test result: If that witness is outside of L^+ then an L-optimum is outside of L^+ by the guide rule and hence DiscardFar can be applied. On the other hand if there is no such witness outside we are going to show that S must contain an L-optimum, allowing for executing DiscardNear.

More formally we define

$$L^+ := \left\{ x \in L \;\middle|\; \begin{array}{l} \text{there is no point } y \in T - F^+ - T_h(x) \\ \text{with } d(x,y) > \alpha \text{ and } w_x(y) \geq w_x(h) \end{array} \right\}$$

as a subset of the leader points.

Place at each point of the boundary ∂L^+ which is not already a node a new temporary node with zero weight. Then we can consider L^+ as a subtree of L. Note that ∂L^+ can contain nodes which are not leaves in L^+. See Figure 2 for an example.

Lemma 2.3 *If $L^+ \neq \emptyset$ then L^+ forms a subtree of S containing the terminal u.*

Proof. No point outside S has an α-neighbor in F^+ and hence $L^+ \subseteq S$. Consider the tree T rooted at h. Let x be a point in L and $x' \in L$ be a descendant of x. If $x \notin L^+$ then there is an $y \in T - F^+ - T_h(x)$ with $d(x,y) > \alpha$ and $w_x(y) \geq w_x(h)$. Clearly $d(x',y) \geq d(x,y)$, $w_{x'}(y) = w_x(y)$, and $y \notin T_h(x') \subseteq T_h(x)$. Therefore $x' \notin L^+$ either. This shows that if a point x is in L^+ then each ancestor of x is in L^+ as well.

Since u is an ancestor of any point in L, the premise $L^+ \neq \emptyset$ implies that $u \in L^+$. Any two points of L^+ are connected via their common ancestor u. □

In the sequel we suppose that ∂L^+ is not empty. The case $\partial L^+ = \emptyset$ turns out to be relatively simple and will be discussed in the proof of Lemma 2.6.

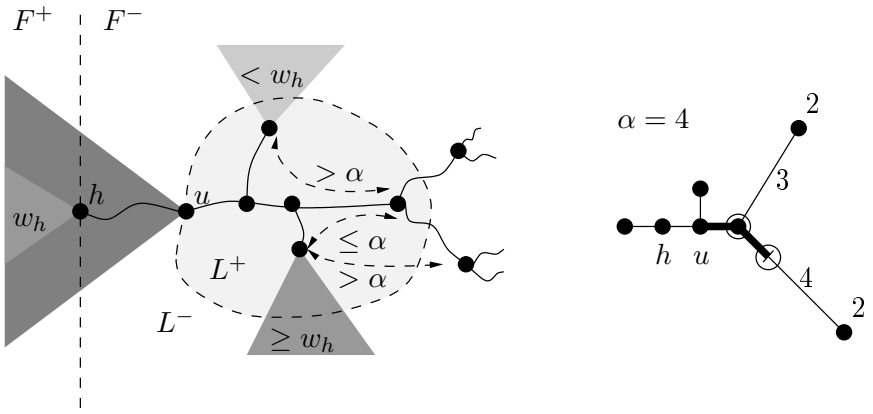

Figure 2: Left: Illustration of set L^+, Right: Example where ∂L^+ contains non-leaves. Unmarked nodes and edges have weight 1. Set L^+ is marked by thick edge segments. Boundary points of L^+ are circled.

Let $x \in \partial L^+$ be a node on the boundary. By splitting at x the tree L decomposes into edge disjoint trees L_i. If $L_i \cap L^+ = \{x\}$ then the tree L_i is called an *x-outer tree* and all of its points are called *x-outer points*.

The node x is called *F^+-independent* if there is an x-outer point y such that $\Phi(y \prec x) > \varphi(h, x)$. This definition is motivated by the property that for F^+-independent nodes there is no reason to search for its witness in the set F^+.

Lemma 2.4 *If $x \in \partial L^+$ is F^+-independent then $T_h(x)$ contains a witness of x as well as an L-optimum.*

Proof. If x is its own witness then both claims are obviously true. Otherwise let y be an α-neighbor of x in $T - T_h(x)$. If $y \in F^+$ then $\Phi(y \prec x) = \varphi(y, x) \leq \varphi(h, x)$. Otherwise $y \in T - F^+ - T_h(x)$ and from $x \in L^+$ it follows that $\Phi(y \prec x) \leq \varphi(h, x)$. Since x is F^+-independent we have $\Phi(x) > \varphi(h, x)$ and therefore y cannot be a witness, hence $T_h(x)$ must contain a witness. The second claim follows directly from the guide rule. □

Lemma 2.5 (Test criterion) *Let the tree L be rooted at u and suppose that $\partial L^+ \neq \emptyset$. If S contains no L-optimal point then there is a unique $x \in \partial L^+$ such that x is F^+-independent but none of its descendants is. Moreover, x has an x-outer α-neighbor as a witness.*

This criterion formalizes the initially described test: If there is an $x \in \partial L^+$ with the property described in the above lemma and where y is its x-outer witness, then either x is already L-optimal or an L-optimum lies outside of L^+. Otherwise S contains an L-optimum. In the former case we call DiscardFar, in the latter DiscardNear is executed.

Proof (of Lemma 2.5). If S has no L-optimal point then all L-optima are contained in one single x-outer tree L' for a suitable $x \in \partial L^+$. By Lemma 2.4 it follows that only

```
1  input terminal tree T with follower set F_u
2  let δ be the median of the multiset { d(u,y) | node y ∈ F_u − u }
3  compute the follower sets   F⁻ := { node y ∈ F_u − u | d(u,y) ≤ δ }
                               F⁺ := { node y ∈ F_u − u | d(u,y) ≥ δ }
4  determine point h in F_u with maximum weight w_u(h) and distance δ to u
5  if h is not a node then turn it into a zero weight node
6  compute ∂L⁺
7  if ∂L⁺ is empty then
8      if δ > α then
9          DiscardFar
10     else
11         DiscardNear
12 else
13     for each x ∈ ∂L⁺ and each x-outer point y
14         if Φ(y ≺ x) > φ(h,x) then mark x
15     unmark all ancestors of marked nodes
16     if ∂L⁺ contains a unique marked node x and
17           x has an x-outer α-neighbor as a witness
18     then DiscardFar and stop
19     else DiscardNear
```

Figure 3: Algorithm HalveFollower(T, F_u)

ancestors of x can be F^+-independent. Moreover x has a witness y in L' by the guide rule.

It remains to show that x is F^+-independent. Let x^* be an L-optimal point in L'. We claim that $\Phi(h \prec x^*) \geq \varphi(h, x)$. This would complete the proof as $\Phi(y \prec x) = \Phi(x) > \Phi(h \prec x^*)$ and y is x-outer. From $x \in S$ it follows that $d(x, h) \leq \alpha$. Therefore x can not prefer x^* over h. This implies $U(x^* \prec h) \subseteq T_h(x)$ and so $w(x^* \prec h) \leq w_h(x)$. On the other hand each point in $T_x(h)$ prefers h over x^* since $d(h, x^*) > \alpha$. Hence $w(h \prec x^*) \geq w_x(h)$. We conclude $\Phi(h \prec x^*) = \varphi(w(h \prec x^*), w(x^* \prec x)) \geq \varphi(h, x)$ as desired. □

Lemma 2.6 (Halve follower tree) *Let T be a terminal tree Φ-equivalent to the input tree T_0, let F_u be its follower tree. Then the algorithm* HalveFollower *constructs a follower tree F'_u with size $|F'_u| \leq \frac{1}{2}|F_u| + 1$ such that replacing F_u by F'_u does not invalidate the Φ-equivalence of T.*

Proof. We claim that algorithm HalveFollower (depicted in Figure 3) performs the construction. We consider four distinct cases.

1. $\partial L^+ = \emptyset$ and $\delta > \alpha$. Let $x \in L$ be an arbitrary leader point. The distance between x and any node in F^+ is greater than α. If the follower places his facility into F^+ we

```
1  set w(h) ← w(F⁻)
2  remove h from F⁻
3  for each node v ∈ F_u − F⁻ adjacent to F⁻
4      connect v to u via a new edge of length d(u, v)
5  discard all nodes of F⁻ from the tree
```

Figure 4: Algorithm DiscardNear to discard most nodes in F^-

```
1  remove h from F⁺
2  for all nodes v of F⁺ (starting with the leaves)
3      let a be the ancestor of v not in F⁺ with maximum distance d(u, a)
4      discard v from the tree and add its weight to a
5  increase the length of the edge incident with h to +∞
```

Figure 5: Algorithm DiscardFar to discard most nodes in F^+

can assume without loss of generality that this location is h. Therefore T remains Φ-equivalent if we change the weight of h to $w_u(h)$, lengthen the connecting edge, and discard all other nodes in F^+ from the tree T. This is exactly what DiscardFar does.

2. $\partial L^+ = \emptyset$ and $\delta \leq \alpha$. Since u is a node contained in L^+, the set L^+ is not empty, and $\partial L^+ = \emptyset$ implies therefore that L is a subset of the α-ball S. The distance between any node in F^- and any leader node $x \in L$ is at most α. Hence the follower has no incentive to choose a node in F^- and discarding these nodes of F_u from the tree does not have any impact to the Φ-scores of points in L.
3. $\partial L^+ \neq \emptyset$ and there is a unique F^+-independent $x \in \partial L^+$ which has no F^+-independent descendants. Moreover this x has an x-outer witness y. Let L' be the x-outer tree containing y. Then either L' contains an L-optimal point by the guide rule. Moreover each point in L' is either equals x when it has witness $y \notin F^+$ or $y \notin L^+$. In the latter case the follower has no incentive to place within the set F^+. Hence T remains Φ-equivalent after the execution of DiscardFar.
4. $\partial L^+ \neq \emptyset$ and there are either zero or at least two F^+-independent nodes x which have no F^+-independent descendants, or x has no x-outer point as a witness. Then S contains L-optimal points by Lemma 2.5 and thus DiscardNear does not affect the Φ-equivalence of T. □

We are now going to show that the running time of HalveFollower is linear in the size of the current terminal tree T. The computation of the median in Line 3 can be carried out in linear time employing the well known median-of-medians algorithm from [BF+73]. We now shed a light on the construction of the boundary set ∂L^+ (Line 6): To this end we describe how to determine the tree L^+. If this is known, the computation of ∂L^+ is easily carried out with an additional depth first search traversal.

1 **input** terminal tree T with terminals u, v and leader tree L
2 let m be the unweighted median of L
3 let m' be the least common ancestor of u, v with respect to the root m
4 through splitting at m and m' the tree L subdivides
5 into edge disjoint components
6 determine witnesses of m and m'
7 **if** $\Phi(z) = \Phi(z \prec z)$ for a $z \in \{m, m'\}$ **then output** optimum z and **stop**
8 the witnesses of m and m' select exactly one
9 of those components, L'
10 choose new terminal nodes s, t from $L' \cap \{u, v, m, m'\}$
11 **if** this only yields one terminal, choose an arbitrary leaf of L'
12 as the second terminal
13 **output** the new terminal tree

Figure 6: Algorithm HalveLeader

For each edge $(s,t) \in L$ we define

$$\delta_s(t) := \sup\{\, d(s,y) \mid y \in T - T_t(s) - F^+,\ w_s(y) \geq w_s(h) \,\}\,. \tag{2}$$

These values can be computed in linear time with the help of two depth first search traversals; the details of a very similar algorithm have been outlined in [NSW07]. To this end we need to discretize the condition "point $y \in T - T_t(s) - F^+$" in (2) to "node $y \in T - T_t(s) - F^+ + h$". Given all those δ-values, the segment of edge (s,t) (the edge is oriented such that $t \in T_h(s)$) which is contained in L^+ is determined by

$$\{\, z \in (s,t) \mid d(t,z) \geq \delta_t(s) - \alpha \,\}$$

which can be computed in constant time.

It remains to argue that all tests whether nodes $x \in \partial L^+$ are F^+-independent (Line 14) can be carried out in total linear time. In order to answer the test for a node $x \in \partial L^+$ it suffices to evaluate $\Phi(y \prec x)$ only for all the α-neighbors of x in all x-outer trees. Since all outer trees are pairwise edge disjoint the total running time for all those tests is linear.

Lemma 2.7 *Algorithm* HalveFollower *runs in linear time $O(|T|)$ on a terminal tree T.* □

2.2 Sparsen the Leader Tree

The second ingredient is a subroutine which decreases the size of the leader tree by almost half of the nodes contained.

Lemma 2.8 (Halve leader tree) *Let T be a terminal tree Φ-equivalent to T_0, let L be its leader tree. Then we can construct in linear time $O(|T|)$ a Φ-equivalent terminal tree T' of size $|T'| = |T|$ which has a leader set L' where $|L'| \leq \frac{1}{2}|L| + 1$.*

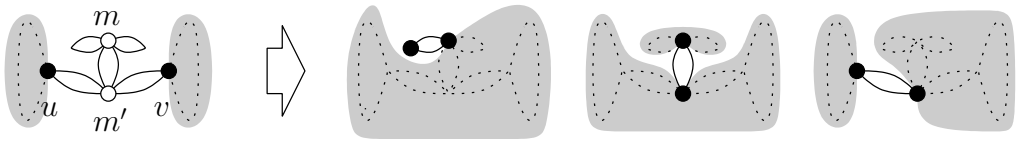

Figure 7: Example of the execution of HalveLeader algorithm. Left: start of phase, right: end of phase. The shaded areas depict the stars of the candies.

Proof. We claim that algorithm HalveLeader depicted in Figure 6 performs the desired construction. The correctness of the algorithm is an immediate consequence of the fact that we restrict the leader set according to the guide rule. For the bound on the running time it suffices to observe that the unweighted median can be determined in linear time employing Goldman's algorithm [Gol71]. □

2.3 The Main Algorithm

We now describe the main algorithm. During the execution the algorithm maintains a current terminal tree T with leader tree L, terminals u, v, and follower trees F_u, F_v. It starts with the input tree T_0 as the initial terminal tree, where two arbitrary leaves play the role of the terminals (thus initially $L = T_0$). Iteratively the algorithm determines the largest among $|L|, |F_u|, |F_v|$ (with a favor for a follower tree in case the maximum is not unique) and halves that component using one of the subroutines HalveFollower or HalveLeader described above. The algorithm ends when the size of T falls below 6 at which time the L-optimum solution in the resulting leader tree will be determined in constant time. The correctness of this algorithm follows from Lemma 2.6 and Lemma 2.8.

We now analyze the running time. To this end we subdivide the sequence of iterations into phases; an operation HalveFollower terminates the current phase. Let n_i denote the size of the terminal tree after the ith phase. Since HalveLeader does not change the terminal tree's size it is easy to see that $n_{i+1} \leq \frac{5}{6} n_i + 1$ for all i. In all but the first phase we can have only one call to HalveLeader: Separating the leader tree of size a at the median yields that at least $\frac{1}{2}a - 1$ nodes are moved into the same follower tree, which then exceeds the size of the new leader tree. (In the first phase we have at most two of these calls.) Therefore the running time of phase i is linear in $O(n_{i-1})$. Since the sequence $(n_i)_i$ is geometrically decreasing this yields an overall linear running time:

Theorem 2.9 *For any monotonous gain function Φ an absolute Φ-solution can be found in time $O(n)$ in a tree with n nodes.*

Corollary 2.10 *An absolute Simpson, security, or Stackelberg solution in a tree can be computed in linear time.* □

3 Conclusions and Further Remarks

In the case of user tolerance parameter $\alpha = 0$ it has been shown in [HTW90] that on a tree the Simpson solution, the security solution, and the Stackelberg solution all coincide with the median and hence can be computed in linear time. In fact this result extends to any monotonous gain function:

Theorem 3.1 *If $\alpha = 0$, for any monotonous gain function Φ the weighted median of a tree is always a Φ-solution.*

Proof. Let m be the weighted median and $x \neq m$. Then

$$\Phi(m) \leq \varphi(\tfrac{1}{2}w(T), \tfrac{1}{2}w(T)) \leq \Phi(m \prec x) \leq \Phi(x)$$

which follows from the property that the weight of any subtree of the forest $T - m$ is at most $\frac{1}{2}w(T)$. □

On the other hand we have given in [SW07] an example which demonstrates that in the case $\alpha > 0$ all those solutions may actually be disjoint. In the current paper we have provided an algorithm for these more general problems which still has linear running time.

Competitor Sensitive Gain Functions

An intrinsic property of the suggested model of monotonous gain functions as specified in Definition 1.3 is that the leader and the follower share the same estimations on the weights of users. An obvious extension of the model is hence to have two different weight functions $w_{\mathrm{L}}, w_{\mathrm{F}}$ for the leader and the follower, respectively, and to define

$$\Phi(y \prec x) = \varphi(w_{\mathrm{F}}(y \prec x), w_{\mathrm{L}}(x \prec y)) .$$

Albeit the resulting function is strictly speaking not a monotonous gain function it can easily be seen that the approach outlined in this paper can be adapted to also handle this extended model.

This can be applied to a generalization of the Stackelberg problem. Normally, when a user is undecided, its demand is split equally amongst the two competing providers [HTW90]. However there can be situations where undecided users split their demand on a per user basis among the two competitors. This can formally modelled by introducing a function $f\colon V \to [0,1]$ which specifies the individual user demand gained by the follower in the case where the user is undecided. (The original Stackelberg problem is then the special case of $f \equiv \frac{1}{2}$.) Thus we end up with a gain function

$$\Phi(y \prec x) := w(y \prec x) + (f \cdot w)(y \sim x) .$$

This can be modelled by a competitor sensitive gain function if we set $w_{\mathrm{F}} := (1 - f) \cdot w$ and $w_{\mathrm{L}} := f \cdot w$ and $\varphi(\mu, \nu) := (f \cdot w)(T) + \mu - \nu$.

Open Problems

It remains an open question how the problem investigated in this report behaves when restricted to nodes only: can the $O(n\,(\log n)^2)$-algorithm from [SW07] for finding a discrete Φ-solution be improved?

References

[BF+73] M. Blum, RW. Floyd, V.R. Pratt, R.L. Rivest, and R.E. Tarjan · *Time bounds for selection* · Journal of Computer and System Sciences (1973), 448–461.

[CM03] C. M. Campos Rodríguez and J. A. Moreno Peréz · *Relaxation of the condorcet and simpson conditions in voting location* · European Journal of Operations Research **145** (2003), 673–683.

[Gol71] A. J. Goldman · *Optimal center location in simple networks* · Transportation Science **5** (1971), 212–221.

[HTW90] P. Hansen, J.-F. Thisse, and R. E. Wendell · *Equilibrium analysis for voting and competitive location problems* · in [MF90], 1990, pp. 479–501.

[MF90] P. B. Mirchandani and R. L. Francis · *Discrete location theory* · Series in Discrete Mathematics and Optimization, Wiley-Interscience, 1990.

[NSW07] H. Noltemeier, J. Spoerhase, and H.-C. Wirth · *Multiple voting location and single voting location on trees* · European Journal of Operations Research **181** (2007), 654–667.

[SW07] J. Spoerhase and H.-C. Wirth · *Relaxed voting and competitive location on trees under monotonous gain functions* · Tech. Report no. 401, University of Würzburg, Department of Computer Science, 2007, `http://www.informatik.uni-wuerzburg.de/forschung/technical_reports/`.