Approximate Maxima Finding of Continuous Functions under Restricted Budget
(Extended abstract)

Evangelos Kranakis [*§] Danny Krizanc [*§] Andrzej Pelc [†§]
David Peleg [‡]

Abstract

A function is distributed among nodes of a graph in a continuous way, i.e., such that the difference between values stored at adjacent nodes is small. The goal is to find a node of maximum value by probing some nodes under a restricted budget. Every node has an associated cost which has to be paid for probing it and a probe reveals the value of the node. If the total budget is too small to allow probing every node, it is impossible to find the maximum value in the worst case. Hence we seek an *Approximate Maxima Finding (AMF)* algorithm that offers the best worst-case guarantee g, i.e., for any continuous distribution of values it finds a node whose value differs from the maximum value by at most g.

Approximate Maxima Finding in graphs is related to a generalization of the multicenter problem and we get new results for this problem as well. For example, we give a polynomial algorithm to find a minimum cost solution for the multicenter problem on a tree, with arbitrary node costs.

[*]Carleton University, School of Computer Science, Ottawa, ON, K1A 5B6, Canada. E-mail: {kranakis,krizanc}@scs.carleton.ca

[†]Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada. E-mail: pelc@uqah.uquebec.ca

[‡]Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: peleg@wisdom.weizmann.ac.il. Part of this research was performed while this author was visiting Carleton University as a COGNOS scholar.

[§]Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grants.

1 Introduction

Suppose that a certain function is distributed among nodes of a graph G in a continuous way, namely, the difference between the function values stored at adjacent nodes is small. A *searcher* situated in one of the graph nodes seeks to find a node of maximum value by *probing* some of the nodes. What limits the scope of the search is the fact that probes are expensive, and the searches operates under a restricted budget. More specifically, every node has an associated *cost* which has to be paid for probing it, and a probe reveals the value of the node. If the total budget is too small to allow probing every node, it is impossible to find the maximum value in the worst case. Hence we seek an *Approximate Maxima Finding (AMF)* algorithm that offers the best *guarantee* g, i.e., for any continuous distribution of values it finds a node whose value differs from the maximum value by at most g.

There are a number of possible schemes of pricing probes, and in what follows we look at the problem of Approximate Maxima Finding under three such schemes. The simplest is the *unit cost* model, where accessing any node costs one unit. The second and most general pricing scheme is based on the assumption that probing different nodes costs different amounts. We refer to this scheme as the *arbitrary cost* model. Finally, in some cases it is plausible that the price is higher for remote nodes (namely, ones far away from the location of the searcher). This is captured by the *distance cost* model, in which the cost of probing a node is proportional to its distance in the graph from the searcher's location h_0.

Example 1 - Least loaded server: Consider a distributed network of servers. The network employs a distributed local load balancing scheme, by which each server periodically compares its load with that of its neighbors, and passes some of the jobs in its queue to certain neighbors, or takes some load off certain other neighbors, according to their relative loads. As a result of this scheme, neighboring servers are nearly balanced, so the load function is more or less continuous on the entire network. However, looking at a three-dimensional "topological map" representing the loads at the various servers, there may still be "hills" at certain regions of the network, representing areas of highly loaded servers, or "valleys" representing areas of lightly loaded servers. Note that as this load function constantly changes, one usually knows the topology of the network but not the current load distribution at any given moment.

Thus a client in need of a free server for an urgent job may still profit from seeking a relatively unloaded server, even if that server is in a remote location on the network. However, the search itself incurs a delay on the job, so it makes sense to limit the search by imposing a budget restriction, and settle for the best server found within the restricted search. In this example, it is probably appropriate to use either the unit cost or the distance cost models, according to the specific characteristics of the underlying communication network at hand. □

Example 2 - WWW access: Our original interest in the above problem was motivated by the following example. Consider the issue of searching the World Wide Web for a particular data. The expanding use of the Internet for commercial purposes makes it increasingly plausible that users will be required to pay for navigating on the Web in the future. This is

already the case to a certain extent, as some sites charge for access to particular Web pages, containing, e.g., on-line magazines or encyclopedias. On the other hand, telecommunication companies that are owners of cables linking sites of the Internet may start charging per use rather than a flat access fee.

Consider an undirected graph whose nodes are Web pages. Two nodes are adjacent if there is a hypertext link between them in at least one direction. Consider a user at a given home page h_0 and consider its connected component G in the above graph. The extremely large number of existing hypertext links joining various pages of the Web guarantee that for most home pages this component G is a large portion of the entire Web. The user searches for data on a particular subject. From the point of view of this task, every Web page of G can be assigned an integer (positive or negative) value representing how closely its content matches the data sought by the user. The home page of the user has value 0. It is reasonable to assume that (in most cases) values of adjacent nodes do not differ much. Indeed, pages represented by those nodes are joined by a hypertext link in at least one direction. Such pages are unlikely to have large difference in values from the point of view of a given data searching task. Suppose, for example, that the user seeks data on *fault-tolerant routing in graphs* and assigns the value v to the Web page *algorithms on graphs*. This page can have hypertext links to *routing* or *fault-tolerant algorithms on graphs*, that will be of slightly larger value for our user, as they approach the investigated subject more closely, or a link to *planarity testing* that is of slightly lesser value, as it deviates from the subject, but is unlikely to have a hypertext link to *Babylonian mythology* that would be clearly of much smaller value than v, although very interesting by itself.

With every Web page we associate a non-negative cost of examining its content. There are several natural ways of pricing such data acquisition. The *unit cost* model can apply in certain cases, and the accesses can be either charged by the telecommunication company for the transmission of data or by the owner of the page. In case of accessing data banks, it may be more natural to assume that different sites will charge differently for their various pages, which justifies using the *arbitrary cost* model. In the case where the price is charged by the telecommunication company it is also plausible that the price will be higher for remote Web pages, hence using the *distance cost* model may be appropriate. □

It turns out that the AMF problem with restricted budget is similar to a generalization of the multicenter problem [3], where a set of k nodes of a graph is sought that minimizes the maximum distance from every node of the graph. Our results imply solutions for some variants of this problem that are of independent interest. For example, we give a polynomial algorithm to find a minimum cost solution for the multicenter problem on a tree, with arbitrary node costs.

1.1 Results of the paper

Most of our results concern the case when the graph G is a chain or, more generally, a tree. For general graphs we show that basic hardness and approximability results can be readily derived by relating the problem to a variant of the *multicenter problem* [3, 4].

For the case of the chain we give formulas indicating which nodes should be probed for the unit and distance cost to get optimal guarantee. For the arbitrary cost we construct a simple optimal algorithm based on shortest paths and give an even faster, linear time approximation, if the costs of all nodes have limited range. In case of arbitrary trees, an optimal polynomial time algorithm for unit cost follows from a result in [3], while we construct an optimal polynomial time algorithm for arbitrary costs. For the distance cost a faster greedy algorithm can be applied. We also remark that AMF in general graphs is an NP-hard problem for all considered ways of pricing.

The paper is organized as follows. In section 2 we formally describe the problem. In section 3 we discuss adaptive versus oblivious algorithms for AMF. We also show the connection of AMF under unit cost to the multicenter problem. Sections 4 and 5 contain AMF algorithms for chains and trees, respectively. Finally, section 6 contains conclusions and open problems.

2 Statement of the Problem

Let $G = (V, E)$ be an undirected graph. For arbitrary nodes x and y, $dist(x, y)$ denotes the length of the shortest path in G joining x and y. Let h_0 be a fixed node. Let $F : V \rightarrow Z$ be an integer valued function representing values of nodes. We assume that $F(h_0) = 0$ and that F is *continuous*, namely, $|F(x) - F(y)| \in \{0, 1\}$, whenever nodes x and y are adjacent. Call a function F satisfying the above properties a *value function* and let $\mathcal{F}$ denote the set of all value functions. For any value function $F \in \mathcal{F}$, $F_{max} = \max(F(v) : v \in V)$. Let $\mathcal{C} : V \rightarrow R^+$ be a function representing the cost of probing nodes of the graph. For a set of nodes $P = \{v_1, ..., v_k\}$, we denote the total cost of P by

$$\mathcal{C}(P) = \sum_{i=1}^{k} \mathcal{C}(v_i).$$

We refer to the constant cost function $\mathcal{C} \equiv 1$ as the *unit cost*, and to the cost function satisfying $\mathcal{C}(x) = dist(x, h_0)$ for all nodes x, as the *distance cost*.

Fix a graph G, a node h_0 and a cost function $\mathcal{C}$. The input of an Approximate Maxima Finding (AMF) algorithm is a positive real B representing the budget allowed for the AMF task and a value function $F \in \mathcal{F}$. The algorithm can *probe* any set of nodes P, whose total cost is within the budget, i.e., such that $\mathcal{C}(P) \leq B$. In the beginning, only the value $F(h_0) = 0$ is known. In the course of the algorithm execution only the values of nodes that have already been probed become known. Decision on which node to probe next can be made dynamically, on the basis of the values discovered so far. For a given value function $F \in \mathcal{F}$ and a given budget B, the output of the algorithm $\mathcal{A}$ is the set $\mathcal{P}(\mathcal{A}, F, B)$ of all nodes that have been probed, together with the node h_0. Among them the node of highest value

$$\mathcal{A}_{max}(F, B) = \max(F(v) : v \in \mathcal{P}(\mathcal{A}, F, B))$$

can be chosen.

For a given AMF algorithm $\mathcal{A}$ and a given input budget B, the *guarantee* of $\mathcal{A}$ for B is defined as

$$\mathcal{G}(\mathcal{A}, B) = \max_{F \in \mathcal{F}}(F_{max} - \mathcal{A}_{max}(F, B)).$$

An AMF algorithm $\mathcal{A}^*$ is *optimal* if, for any budget B and any AMF algorithm $\mathcal{A}$, $\mathcal{G}(\mathcal{A}^*, B) \leq \mathcal{G}(\mathcal{A}, B)$. Let

$$\mathrm{Rad}(\mathcal{A}, F, B) = \max_{v \in V} \min_{u \in \mathcal{P}(\mathcal{A}, F, B)}(dist(u, v)),$$

and let

$$\mathrm{Rad}(\mathcal{A}, B) = \max_{F \in \mathcal{F}}(\mathrm{Rad}(\mathcal{A}, F, B)).$$

Note that these two "radius parameters" are dependent on $\mathcal{A}$ and F, in that the choice of the probe set $\mathcal{P}(\mathcal{A}, F, B)$ can be made by $\mathcal{A}$ dynamically on the basis of the F values discovered along the way. In contrast, one can define a *static* radius parameter, based only on the graph G and the budget B, as follows: For a set of nodes S, let

$$\mathrm{Rad}(S) = \max_{v \in V} \min_{u \in S}(dist(u, v)).$$

Let $\mathcal{S}$ be the collection of all sets $S \subseteq V$ whose total cost is within the budget, i.e., such that $\mathcal{C}(S) \leq B$. Now let

$$\mathrm{Rad}(B) = \min_{S \in \mathcal{S}}(\mathrm{Rad}(S)).$$

In the sequel we rely on the following central observation, based on the continuity of the value functions.

Lemma 2.1 *For every algorithm $\mathcal{A}$ and budget B, $\mathcal{G}(\mathcal{A}, B) = \mathrm{Rad}(\mathcal{A}, B)$.* □

As a consequence, an optimal AMF algorithm $\mathcal{A}^*$ can be thought of as an algorithm minimizing $\mathrm{Rad}(\mathcal{A}, B)$ for any budget B.

3 Adaptive vs. Oblivious Algorithms

Since values of nodes are not known *a priori*, there are two natural classes of AMF algorithms. An *oblivious* algorithm is required to decide on the entire set of probed nodes at once, while an *adaptive* algorithm can probe nodes one by one, computing the next node to probe on the basis of values of nodes already probed. At first glance, adaptive algorithms are more flexible and thus should be able to provide better guarantees. While this is likely to be true on average, we have the following simple result showing that in the *worst case*, adaptivity does not help in Approximate Maxima Finding.

Proposition 3.1 *For any given graph G, node h_0 and cost function $\mathcal{C}$, there exists an oblivious AMF algorithm $\mathcal{A}^{ob}$ which is optimal (among all AMF algorithms, including adaptive ones).*

PROOF Let $\mathcal{A}^*$ be an optimal AMF algorithm. Fix an input budget B and let $P = \{h_0, v_1, ..., v_k\}$ be the output of $\mathcal{A}^*$ for the value function $F \equiv 0$. We claim that the oblivious algorithm $\mathcal{A}^{ob}$ that outputs the set P for input B (regardless of the value function) is also optimal. We have to show that, for any B and any $\mathcal{A}$,

$$\max_{F \in \mathcal{F}}(F_{max} - \mathcal{A}^{ob}_{max}(F, B)) \leq$$

$$\max_{F \in \mathcal{F}}(F_{max} - \mathcal{A}_{max}(F, B)).$$

Fix B. Let $S = \mathcal{A}^{ob}(F, B)$ and $r = \mathrm{Rad}(\mathcal{A}^{ob}, F, B)$ (regardless of $F \in \mathcal{F}$). Take any function $F \in \mathcal{F}$. We have

$$F_{max} - \mathcal{A}^{ob}_{max}(F, B) \leq \mathrm{Rad}(\mathcal{A}^{ob}, F, B) = r.$$

Hence, for any AMF algorithm $\mathcal{A}$ it suffices to show a value function $F \in \mathcal{F}$, such that $r \leq F_{max} - \mathcal{A}_{max}(F, B)$. Let Φ be a value function such that $\Phi(v) = 0$, for all $v \in S$ and $\Phi(v) = r$, for some $v \in V$. Such a function exists by definition of r. By definition of S, $\mathcal{A}^*_{max}(\Phi, B) = 0$. Since $\mathcal{A}^*$ was optimal, $\mathcal{A}(\Phi, B) \leq 0$, for any $\mathcal{A}$ and hence $\Phi_{max} - \mathcal{A}_{max}(\Phi, B) \geq r$. □

The idea behind the above proof further reveals the following.

Proposition 3.2 *For any given graph G, node h_0 and cost function $\mathcal{C}$, the guarantee given by the (oblivious) optimal AMF algorithm $\mathcal{A}^*$ satisfies $\mathcal{G}(\mathcal{A}^*, B) = \mathrm{Rad}(B)$.* □

In view of the above proposition, in the rest of the paper we will restrict attention to non-adaptive algorithms. It is now easy to notice the similarity of unit cost Approximate Maxima Finding to the *multicenter problem* [3]. In the latter problem, the input is a graph G and a positive integer k. The goal is to find a set S of k nodes that minimizes $\max_{v \in V} \min_{u \in S}(dist(u, v))$. Hence finding the optimal AMF algorithm for a graph G, node h_0, unit cost function $\mathcal{C}$ and budget k is equivalent to solve the multicenter problem for G and $k+1$ with the additional restriction that h_0 be in the multicenter S. Both problems are polynomially equivalent.

4 Algorithms for Chains

In this section we present optimal AMF algorithms for the $(n+1)$-node chain with nodes $0, 1, ..., n$.

4.1 Unit cost function

For the unit cost function, optimal AMF for the chain is given by the following proposition. We omit the easy proof.

Proposition 4.1 *Let G be the $(n+1)$-node chain and h_0 its end-point 0. Then the optimal AMF algorithm with budget k has guarantee $g = \lceil \frac{n}{2k+1} \rceil$ and consists of probing nodes $2g$,$4g$,... $2kg$.* □

4.2 Distance cost function

For the distance cost function we also obtain a formula indicating which nodes should be probed by an optimal algorithm.

Proposition 4.2 *Let G be the $(n+1)$-node chain and h_0 its end-point 0. Let $P_k = n\frac{k(k+1)}{2k+1}$. Suppose that the budget B satisfies $P_{k-1} < B \leq P_k$, for some $k = 1, 2, ..., n-1$. Then the optimal AMF algorithm with budget B has guarantee $g = \lceil \frac{n}{k} - \frac{B}{k^2} \rceil$ and consists of probing nodes $n-g$, $n-3g$,..., $n-(2k-1)g$.*

PROOF First note that since $B > P_{k-1} = n\frac{k(k-1)}{2k-1}$, we have $(2k-1)g < n$ and hence all proposed probes are feasible (belong to the set $\{1, ..., n\}$). On the other hand,

$$\sum_{i=1}^{k}(n-(2i-1)g) \leq B,$$

hence the proposed probes are within the budget. Since distances between consecutive probes are $2g$ and the probe $n-g$ is at distance g from the node n, in order to show that our algorithm has guarantee g it is enough to show that the distance between $h_0 = 0$ and the closest probe is at most $2g$. We have

$$g \geq \frac{n}{k} - \frac{B}{k^2} \geq \frac{n}{k} - \frac{P_k}{k^2} = \frac{n}{2k+1},$$

which implies $n-(2k-1)g \leq 2g$, as needed.

Suppose that some other algorithm gives a better guarantee $g' < g$ for the same budget B. Thus the largest probed node must be larger than $n-g$, the second largest must be larger than $n-3g$, etc., the smallest must be larger than $n-(2k-1)g$. Thus the total cost must be larger than

$$\sum_{i=1}^{k}\left(n-(2i-1)\left(\frac{n}{k}-\frac{B}{k^2}\right)\right) = B,$$

contradiction. □

4.3 Arbitrary cost function

Let B be the given budget and g the desired guarantee. Assume that the cost of probing node i is given by $c(i), i \leq n$. For a given budget we consider the problem of moving from node 0 to node n such that the total cost of visited nodes does not exceed the given budget B.

Theorem 4.1 *There is an $O(n^2 \log n)$ algorithm which for a given chain with nodes $0, 1, \ldots, n$ and budget B determines a set of probes having the smallest possible guarantee g and not exceeding the given budget.*

PROOF We transform the problem into a shortest path problem as follows. Fix g. For each node i add the $m = \min\{g, n-i\}$ directed edges

$$(i, i+1), (i, i+2), \ldots, (i, i+m).$$

Each of these edges is given weight 0. Now replace each node i with a directed edge of weight $c(i)$. The above problem is now transformed into the problem of finding a path from 0 to n which has total weight at most B. This problem is easily solved using the modified Dijkstra's algorithm (e.g. see [1]) whose running time is $O(n^2)$.

For the given budget B we execute binary search on the guarantee g, $(g \leq n)$. The resulting algorithm has running time $O(n^2 \log n)$. □

Remark: In cases where the complexity of the above algorithm is of concern, one may be willing to settle for *weaker* guarantees, dependent on the range of possible costs, and get a faster algorithm for determining the probe set. In particular, if it is known that the costs of the nodes are taken from a given range $[1, X]$ for some sufficiently small X, then it is possible to set

$$k = \lfloor B/X \rfloor,$$

and employ the rule of Subsect. 4.1. For this rule we can show the following.

Theorem 4.2 *For a given chain with nodes 0, 1,..., n, for costs taken from the range* $[1, X]$ *and budget* B*, the above explicit rule determines a set of probes not exceeding the given budget, and having a guarantee at most* Xg*, where* g *is the smallest possible guarantee.* □

5 Algorithms for Trees

In this section we study the problem assuming the graph G at hand is a tree. For trees, if the model is based on a unit cost function, then by the observation made at the end of Sect. 3, the problem can be solved in polynomial time by employing a simple variant of the algorithm of [3] for the multicenter problem. Hence in the remainder of this section we consider the distance cost function and the case of arbitrary costs.

5.1 Distance cost function

In this subsection we give an algorithm for the case where $\mathcal{C}$ is the distance cost function and the graph is a tree.

We are given a tree T and a budget B. For a candidate value g, Algorithm $\mathcal{A}_g$ returns a (minimum cost) set $\mathcal{P}$ of probes with radius $\mathrm{Rad}(\mathcal{P}) = g$. It follows from Prop. 3.2 that the lowest value of g for which the cost of the solution produced by the algorithm fits the given budget B is the optimal guarantee.

Algorithm $\mathcal{A}_g$:

1. $\mathcal{P} \leftarrow \{h_0\}$;
2. mark the root h_0 and every node v such that $\mathrm{Depth}(v) \leq g$ as **ok**.
3. **while** T still contains unmarked nodes **do**
   (a) Let l be a deepest unmarked node;
   (b) Let c be the g-th ancestor of l;
   (c) $\mathcal{P} \leftarrow \mathcal{P} \cup \{c\}$;
   (d) Mark every node v of distance at most g from c as **ok**;

Let $\mathcal{P} = \{c_1, c_2, \ldots, c_k\}$ be the set of probes and $L = \{l_1, l_2, \ldots, l_k\}$ the set of *witnesses* selected in step 3(a). We prove the following lemmas.

Lemma 5.1 *For $i < j$, the nodes l_i and l_j are at distance at least $2g$.*

PROOF The node l_i is at distance g from c_i. Since $i < j$, l_j cannot be marked **ok** during the execution of the algorithm at node c_i. Therefore it must be the case that l_j is marked **ok** at the execution of the algorithm at c_j. Since the topology is a tree it is clear that l_i and l_j are at distance at least $2g$. □

Lemma 5.2 *For any set $\mathcal{P}'$ of probes such that* $\mathrm{Rad}(\mathcal{P}') = g$,

$$\mathcal{C}(\mathcal{P}) \leq \mathcal{C}(\mathcal{P}').$$

PROOF Let $\mathcal{P}'$ be a set of probes with radius $\mathrm{Rad}(\mathcal{P}') = g$. For every i, let $c'_i \in \mathcal{P}'$ be the probe within distance at most g of l_i. By Lemma 5.1 the probes c'_i are pairwise distinct. By the choice of the probes c_i and c'_i,

$$\mathrm{Depth}(c'_i) \geq \mathrm{Depth}(l_i) - g = \mathrm{Depth}(c_i).$$

It follows that

$$\mathcal{C}(\mathcal{P}') \geq \sum_{i=1}^{k} \mathrm{Depth}(c'_i) \geq \sum_{i=1}^{k} \mathrm{Depth}(c_i) = \mathcal{C}(\mathcal{P}).$$

The lemma follows. □

We can now prove the following theorem.

Theorem 5.1 *There is an $O(n \log d)$ algorithm which for a given tree of size n and depth d, and budget B determines a set of probes having the smallest possible guarantee g and not exceeding the given budget.*

PROOF Given a tree T and a budget B we execute binary search on the possible values of g in order to find the smallest g and a set $\mathcal{P}$ of probes such that $\mathcal{C}(\mathcal{P}) \leq B$, and $\mathrm{Rad}(\mathcal{P}) = g$. To complete the proof we argue that $\mathrm{Rad}(B) = g$. The theorem then follows by Prop. 3.2.

To see why $\mathrm{Rad}(B) = g$, note that had there existed a smaller value $g' < g$ and a set of probes $\mathcal{P}'$ such that $\mathcal{C}(\mathcal{P}') \leq B$ and $\mathrm{Rad}(\mathcal{P}') = g'$, then when invoking algorithm $\mathcal{A}$ with parameter g', the resulting minimum cost solution $\mathcal{P}''$ should have cost no more than B, and our search should have ended up with g' instead of g; contradiction. □

5.2 Arbitrary costs

In this section we give an optimal algorithm for the case where $\mathcal{C}$ is an arbitrary cost function and the graph is a tree rooted at the node h_0.

Definition 5.1 *For any tree T, rooted at a node r, a set of probes, $v_1, \ldots, v_k$* achieves guarantee g with surplus s *($-g \leq s \leq g$) if for all $v \in T$ such that $dist(v,r) \geq -s$, there exists an i such that $dist(v_i, v) \leq g$ and if $s \geq 0$ there exists an i such that $dist(v_i, r) \leq g - s$.*

Let $X_g^s(r)$ be the cost of the minimum cost set of probes that achieves guarantee g with surplus s for a tree with root r. Then we have the following lemma:

Lemma 5.3 *Let T be a tree with root r where r has children $r_1, \ldots, r_d$.*
If $d > 0$ then

$$X_g^s(r) = \begin{cases} \mathcal{C}(r) + \sum_{j=1}^{d} X_g^{-g}(r_j), & \text{if } s = g \\ \min\{X_g^{s+1}, \quad \min_i\{X_g^{s+1}(r_i) + \sum_{j \neq i} X_g^{-s}(r_j)\}\}, & \text{if } 0 \leq s < g \\ \min\{X_g^{s+1}, \sum_{j=1}^{d} X_g^{s+1}(r_j)\}, & \text{if } -g \leq s < 0. \end{cases}$$

If $d = 0$ then

$$X_g^s(r) = \begin{cases} \mathcal{C}(r), & \text{if } 0 \leq s \leq g \\ 0, & \text{if } g \leq s < 0. \end{cases}$$

PROOF (Sketch.) Consider the case $s = g$. For a set of probes to achieve guarantee g with surplus g the root must be chosen. Once the root has been chosen the subtrees only require surplus $-g$. Using the fact that $X_g^s(r) \leq X_g^{s+1}(r)$ for $-g \leq s < g$ (since a set of probes with surplus $s+1$ is also a set of nodes with surplus s), the formula follows. The other cases are similar. □

Theorem 5.2 *For any n node tree T (with root r) and cost function $\mathcal{C}$ there exists an optimal (oblivious) AMF algorithm with running time $O(n^2 \log n)$.*

PROOF Using Lemma 5.3 one can devise a dynamic programming algorithm to compute $X_g^g(r)$ starting at the leaves and moving up the tree to the root. By definition $X_g^g(r)$ is equal to the minimum budget B required to achieve a guarantee g on the given tree with $h_0 = r$. Using binary search on g, one can compute the maximum g that can be achieved for a certain budget B. It is straightforward to adapt the algorithm to compute the required probe set. The running time of the dynamic programming algorithm is $O(\sum_{v \in T} g \cdot deg(v)) = O(n^2)$ and it is applied $O(\log n)$ times. □

Note that the dynamic programming algorithm alluded to above can easily be adapted to solve the problem of finding a minimum cost solution to the multicenter problem on a tree, where the nodes are assigned arbitrary weights.

6 Conclusion

6.1 Arbitrary graphs

For arbitrary graphs, the problem of optimizing the guarantee under a given budget becomes NP-hard. This has been shown for the unit cost multicenter problem in [3], and a simple variation gives the same result for AMF under the unit cost model, hence it is clearly applicable also for the arbitrary cost model. It is also straightforward to establish hardness for the distance cost model, say, via a reduction from Exact 3-Cover (X3C). In summary we have:

Proposition 6.1 *The AMF problem for arbitrary graphs is NP-hard under the unit cost, distance cost and arbitrary cost models.* □

As for approximations, straightforward variants of the approximation algorithms of [4] for the multicenter problem yield also approximation algorithms for the unit cost model and the arbitrary cost model. The latter naturally encompasses also the distance cost model, with the same performance guarantees. Hence we have:

Proposition 6.2 *There exists an approximation algorithm for the AMF problem on arbitrary graphs*

1. *with approximation ratio 2 under the unit cost model, and*
2. *with approximation ratio 3 under the distance cost or the arbitrary cost models.* □

6.2 Directions for future research

Our results all apply to the worst-case behavior of the problem, and are restricted to deterministic AMF algorithms. It may be both of theoretical interest and practical significance to study *average case* behavior of Approximate Maxima Finding (assuming some random distribution of the value and/or cost functions), as well as to develop randomized algorithms for the problem. In both cases, it is anticipated that the equivalence between adaptive and oblivious strategies will disappear, and adaptive algorithms will perform better.

Another interesting and potentially realistic variant of the problem is based on assuming that the cost of probing a node is somehow related to its quality, hence also to the probability of obtaining a good solution by probing it.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms", Electrical Engineering and Computer Science Series, M.I.T. Press, 1990.

[2] S. L. Hakimi, E. F. Schmeichel, and M. Labbé, "On Locating Path- or Tree-Shaped Facilities on Networks", Networks, Vol. 23 (1993) 543 - 555.

[3] O. Kariv and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems. I: the p-Centers", SIAM J. Appl. Math., Vol. 37, No 3, Dec. 1979.

[4] D. S. Hochbaum and D. B. Shmoys, "Powers of Graphs: A Powerful Approximation Technique for Bottleneck Problems", in Proceedings of STOC 1984, pp. 324 - 333.