Improved Approximation Algorithms for the Freeze-Tag Problem

Esther M. Arkin
Applied Math & Statistics
Stony Brook University
Stony Brook, NY 11794-3600
estie@ams.sunysb.edu

Michael A. Bender
Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
bender@cs.sunysb.edu

Dongdong Ge
Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
dge@cs.sunysb.edu

Simai He
Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
simaihe@cs.sunysb.edu

Joseph S. B. Mitchell
Applied Math & Statistics
Stony Brook University
Stony Brook, NY 11794-3600
jsbm@ams.sunysb.edu

ABSTRACT

In the Freeze-Tag Problem, the objective is to awaken a set of "asleep" robots, starting with only one "awake" robot. A robot awakens a sleeping robot by moving to the sleeping robot's position. When a robot awakens, it is available to assist in awakening other slumbering robots. The objective is to compute an optimal *awakening schedule* such that all robots are awake by time t^*, for the smallest possible value of t^*. Because of its resemblance to the children's game of freeze-tag, this problem has been called *Freeze-Tag Problem (FTP)*.

A particularly intriguing aspect of the FTP is that any algorithm that is not purposely unproductive yields an $O(\log n)$-approximation, while no $o(\log n)$-approximation algorithms are known for general metric spaces.

This paper presents an $O(1)$-approximation algorithm for the FTP in unweighted graphs, in which there is one asleep robot at each node. We show that this version of the FTP is NP-hard.

We generalize our methods to the case in which there are multiple robots at each node and edges are unweighted; we obtain a $\Theta(\sqrt{\log n})$-approximation in this case. In the case of weighted edges, our methods yield an $O((L/d)\log n)$-approximation algorithm, where L is the length of the longest edge and d is the diameter of the graph.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures, geometrical problems and computations, sequencing and scheduling*; G.2.2 [**Discrete Mathematics**]: Graph Theory

General Terms

Algorithms, Theory

Keywords

Freeze-tag problem, TSP, data dissemination, multicast problem, minimum broadcast-time problem, minimum gossip time problem, network optimization, swarm robotics, scheduling, approximation algorithms, NP-hardness

1. INTRODUCTION

The Freeze-Tag Problem (FTP) [5] is a parallel version of the Traveling Salesman Problem (TSP) that naturally arises in the field of *swarm robotics*. In the FTP we have *n robots*, which are located at points in some metric space (e.g., the vertices of an edge-weighted graph). Initially, there is one *awake* or *active* robot and all other robots are *asleep*, i.e., in a *stand-by mode*. Our objective is to "wake up" all of the robots as quickly as possible. In order for an active robot to awaken a sleeping robot, the awake robot must travel to the location of the slumbering robot. Once awake, this new robot is available to assist in rousing other robots. The objective is to minimize the *makespan*, that is, the time when the last robot awakens. The FTP can be expressed as a combinatorial optimization problem as follows: Given a set of points in metric space, find an arboresence (*awakening tree*) of minimum height such that each node has out-degree at most two. The FTP derives its name because this problem is reminiscent of the children's game of "freeze-tag."

The Freeze-Tag Problem is seen to be a parallel version of the Traveling Salesman Problem, in which the cities of the TSP instance correspond to the asleep robots' initial locations in the FTP, and the objective is to visit all cities as rapidly as possible. The FTP has the additional feature that whenever a salesman visits a city he may also recruit other salesmen in that city to help visit other cities, in parallel.

SPAA'03, June 7–9, 2003, San Diego, California, USA.

In addition to being a parallel version of the TSP, the Freeze-Tag Problem is also a natural hybrid of problems in broadcasting, routing, scheduling, and network design. The FTP is a *network design* problem because the optimal schedule is determined by a spanning binary tree of minimum depth in a weighted graph. As in *broadcasting problems*, the goal is to disseminate information in a network. The FTP has elements of *routing*, since robots must travel to awaken others or to transfer information. Finally, the FTP has elements of *scheduling*, with the number of processors increasing over time, and scheduling techniques are often relevant [5].

Other applications of the FTP arise in the context of distributing data (or some other commodity), where physical proximity is required for transmittal. Proximity may be required because wireless communication is too costly in terms of bandwidth or because there is too much of a security risk. Solutions to the FTP determine how to propagate the data to the entire set of participants in the most efficient manner.

Related Work. What makes the FTP particularly intriguing is that any nonlazy strategy yields an $O(\log n)$-approximation (Proposition 1.1 of [5]), while no strategy is known for general metric spaces that yields an $o(\log n)$-approximation. Arkin et al. [5] showed that even simple versions of the problem (e.g., in star metrics) are NP-complete. They give an efficient polynomial-time approximation scheme (PTAS) for geometric instances on a set of points in any constant dimension δ. They also give a variety of results for star metrics (including an $O(1)$-approximation) and for ultrametrics, for which an $o(\log n)$-approximation is possible.

Sztainberg et al. [26] analyze and implement heuristics for the FTP. They show that the greedy strategy gives a tight approximation bound of $\Theta(\sqrt{\log n})$ for the case of points in the plane and, more generally, greedy yields a $\Theta((\log n)^{1-1/\delta})$-approximation for points in $\mathbb{R}^\delta$. They also present experimental results on classes of randomly generated data, as well as on datasets from the TSPLIB [22].

There is an abundance of prior work on the dissemination of data in a graph. Most closely related to the FTP are the *minimum broadcast time problem*, the *multicast problem*, and the related *minimum gossip time problem*. See [17] for a survey; see [7, 20] for recent approximation results. However, the proximity required in the FTP leads to significant differences: While the broadcast problem can be solved in polynomial time in tree networks, the FTP turns out to be NP-hard on the seemingly easy class of weighted stars [5].

In the field of robotics, several related algorithmic problems have been studied for controlling swarms of robots to perform various tasks, including environment exploration [1, 2, 8, 9, 10, 11, 16, 19, 28, 30], robot formation [23, 24, 25], searching [29], and recruitment [27]. Ant behaviors have inspired algorithms for multi-agent problems such as searching and covering; see, e.g., [27, 28, 29]. Multi-robot formation in continuous and grid environments has been studied recently by several researchers; see, e.g., [12, 24, 25]. The objective is for distributed robots to form shapes such as circles of a given diameter, lines, etc. without using global control.

Gage [13, 14, 15] has proposed the development of command and control tools for arbitrarily large swarms of microrobots. He originally posed to us the problem of how to "turn on" a large swarm of robots efficiently; this question is modeled here as the FTP.

Another related problem is to consider variants in which all robots are mobile, but they still have to meet in order to distribute important information. The two-robot scenario with initial positions unknown to both players is the problem of rendezvous search that has received quite a bit of attention; see [4, 21] and the forthcoming book by Alpern and Gal [3] for an overview.

Preliminaries. Let $R = \{v_0, v_1, \ldots, v_{n-1}\} \subset \mathcal{D}$ be a set of n robots in a domain $\mathcal{D}$. We assume that the robot at v_0 is the *source robot*, which is initially awake; all other robots are initially asleep. We let $d(u,v)$ denote the distance between two points, $u, v \in \mathcal{D}$.

The domain $\mathcal{D}$ is specified by a graph $G = (V, E)$. We let $d = \max_{u,v\in\mathcal{D}} d(u,v)$ denote the *diameter* of the graph. The graph G is *unweighted* if all edges have the same length (without loss of generality, unit length) and there is exactly one robot at each node. We say the graph has *unweighted edges and weighted nodes* if all edges have the same length but there may be multiple asleep robots initially placed at nodes. For graphs with weighted edges, we let L denote the length of the longest edge.

Summary of Results.

1. We give a "Density-Based" strategy for the FTP based on computing a dense region of the domain to awaken first. This approach leads to a simple $O(1)$-approximation for Euclidean spaces and an $O((\log n)^{\log(5/3)})$-approximation algorithm for unweighted graphs. While this strategy improves upon previous results, its approximation factor is not as good as the one stated below; it is, however, a simple, promising approach that may lead to an improved solution for general graphs.

2. We give a "Sibling-Based" strategy that yields the following new results:

   (a) An $O(1)$-approximation for unweighted graphs.

   (b) An $O(\sqrt{\log n})$-approximation for graphs with unweighted edges and weighted nodes.

   (c) An analysis of the makespan when the strategy is applied to general graphs, having weighted edges and weighted nodes. We show that the makespan is $O(d + L\log n)$, resulting in an $O((L/d)\log n)$-approximation algorithm.

   (d) A proof that the strategy gives an $O(1)$-approximation for computing an awakening that minimizes the total distance traveled by all robots in a general tree.

3. We explore the limitation of the two proposed methods. Both strategies begin by finding an arbitrary approximate minimum-height spanning tree, where the edges in the spanning tree are the *only* edges that the robots ever traverses. We prove that without a better method for finding such spanning trees, we cannot achieve an approximation ratio better than $O(\sqrt{\log n})$, even for graphs with unweighted edges and weighted nodes.

4. We prove that the FTP is NP-hard in unweighted graphs.

2. A DENSITY-BASED ALGORITHM

The main idea of this density-based strategy is to define a dense region of the domain and to awaken the robots in this region first. The subtlety of this approach lies in the definition of what it means for a region to be "dense".

2.1 Euclidean Space

If G is a Euclidean graph in $\mathbb{R}^2$, then the density-based strategy yields a $(3+\varepsilon)$-approximation algorithm. Recall that d is the diameter of the graph, and therefore that all robots are in a d-by-d region of the plane. Conceptually, we divide the plane into $\sqrt{n}$ squares, each of size $d/n^{1/4}$-by-$d/n^{1/4}$. We send the first robot to the densest square, i.e., the square with the most robots. By a simple counting argument, this square contains at least $\sqrt{n}$ robots. We awaken this square using any nonlazy strategy, which costs at most $O(d \log n/n^{1/4})$. Now we have at least $\sqrt{n}$ robots, and we use one robot to begin awakening the robots in each other square, in parallel, which again costs $O(d \log n/n^{1/4})$. We conclude:

THEOREM 1. *The density-based strategy in $\mathbb{R}^2$ is a $(3 + o(1))$-approximation algorithm. In fact, it is a $(3 + o(1))$-approximation algorithm for any dimension that is $o(\log n/ \log \log n)$.*

PROOF. The time for the robot to enter the densest square is at most OPT, and the time for each of these robots to reach a nonempty square is at most 2OPT. The time to awaken the robots in each square is $O(d \log n/n^{1/4}) = O(\text{OPT}\cdot\log n/n^{1/4})$, which is a low-order term. The proof is similar in higher dimensions and is omitted in this abstract. $\square$

In summary, the density-based algorithm yields a simpler $O(1)$-approximation algorithm than previous methods [5], and it applies to higher dimensions.

2.2 Unweighted Graphs

In this section we prove the following theorem for the FTP in unweighted graphs.

THEOREM 2. *There exists an $O((\log n)^{\log(5/3)})$-approximation algorithm for the FTP on unweighted graphs.*

Lower Bound. We begin by proving the following lower bound on the makespan for unweighted graphs.

LEMMA 1. *For an unweighted graph $G = (V, E)$, there is a lower bound $max(\log n, d/2)$ on the optimal awakening time* OPT.

PROOF. In each time step each (awake) robot can awaken at most one other robot. Therefore at time i, at most 2^i robots are awakened. The lower bound of $\log n$ follows.

The awakening signal needs to travel from the starting node v_0 to the farthest node in G. This distance is at least half of the diameter d, implying that $d/2$ is a lower bound for OPT. $\square$

Dense Subtrees. Let $h(T)$ denote the height of tree T, and let $n = N(T)$ denote the number of nodes in T. We say that a subtree T' of T is *dense* if $h(T') \leq \frac{1}{2}(h(T) + 1)$ and $N(T') \geq \sqrt{n}$.

LEMMA 2. *For any tree T there exists a dense subtree T', which can be found in $O(n)$ time.*

PROOF. Cut the tree at half of its height, so that the leaves of the top subtree are the roots of the bottom subtrees. If the top subtree is dense, then we are done. If not, then the number of nodes in the top subtree is less than $\sqrt{n}$, implying that the number of leaves of the top subtree is less than $\sqrt{n}-1$. (The 1 is subtracted because the root of the top subtree is not a leaf.) Therefore there are less than $\sqrt{n}-1$ bottom subtrees. There are more than $n-\sqrt{n}$ nodes in the bottom subtrees, so there exists at least one bottom subtree with at least $\sqrt{n}$ nodes. Because the heights of the subtrees satisfy the constraint, this bottom subtree is dense.

A dense subtree is readily found in time $O(n)$ using depth-first search to label nodes with their heights and number of descendants, and then checking the number of descendants for the nodes at half the height. $\square$

Once we have a subtree T' with at least $\sqrt{n}$ nodes, we can assume T' has exactly $\lceil\sqrt{n}\rceil$ nodes by repeatedly removing leaves.

Density-Based Algorithm.

1. We first find a spanning tree T of graph G rooted at v_0, whose height is at most d (e.g., using breadth-first search). Define h to be the height of tree T, which is between $d/2$ and d. The awakening algorithm will now only use the edges in T.
2. Identify a dense subtree T' and activate it recursively. The base case is $h = 1$, for which greedy is trivially an optimum strategy.
3. Activate $2\lceil\sqrt{n}\rceil$ leaves in T by sending each of those activated robots from T' to any two leaves. Awakening these leaves takes at most $4h$ time. If there are not enough leaves (which means all the leaves have been activated by this time), the whole tree has already been activated, since the awakening signal must travel through all the nodes to reach all the leaves. After these leaves are activated, remove them from the tree T.
4. Return the T' robots back to their original positions (this preserves the connected tree structure), unless they are leaf nodes of T, in which case we remove them from the tree.
5. Divide the tree T into at most $2\sqrt{n}$ subtrees each containing at most $\sqrt{n}$ nodes. (The subtrees-finding algorithm appears below.)
6. Each leaf ℓ_i from step 3 awakens the ith subtree recursively.

   Remark: We do not use the robot on the root node of a subtree in the awakening process of that subtree, since this robot may be needed in another subtree. Instead, we use the robot that has been sent to the root node to do the recursive awakening process. Thus, we do not affect the structure of the other subtrees, which might contain this node.

Subtrees Finding Algorithm. Now we give the subtree-finding algorithm, which determines a set of at most $2\sqrt{n}$ subtrees covering T, each having at most $\sqrt{n}$ nodes.

LEMMA 3. *For any tree T having n nodes, there exist subtrees $T_1, T_2, \ldots$ such that: (1) each subtree T_i contains at most $\sqrt{n}$ nodes; (2) there are at most $2\sqrt{n}$ subtrees; and (3) if two subtrees are not node disjoint, then they share precisely one node, and this common node is a root of one of the two subtrees.*

PROOF. Let T be any tree having at least $\sqrt{n}$ nodes. Our proof is based on showing the existence of a subtree, $\hat{T}$, of T that has the properties that (a) the number of nodes in $\hat{T}$ is between $\sqrt{n}/2$ and $\sqrt{n}$, i.e., $\sqrt{n}/2 \le N(\hat{T}) \le \sqrt{n}$.; and (b) if node $u \in \hat{T}$ is not the root of $\hat{T}$, then all of u's descendants in T are also in $\hat{T}$.

For any node v, let $w(v)$ denote the number of descendants of v in T, including v itself. Let u be a node of T of minimum height such that $w(u) > \sqrt{n}$. Let the children of u be denoted by $v_1, v_2, \cdots, v_k$. By our choice of u, we know that $w(v_j) \le \sqrt{n}, j = 1, 2, \cdots, k$. We construct the subtree $\hat{T}$ as follows: If there exists a node v_i such that $\sqrt{n}/2 \le w(v_i) \le \sqrt{n}$, then choose this v_i and all of its descendants to be $\hat{T}$. Otherwise, if no such v_i exists, then we let u be the root of the tree $\hat{T}$, and then add to $\hat{T}$, successively, the subtrees rooted at $v_1, v_2, \ldots$, until the tree $\hat{T}$ has at least $\sqrt{n}/2$ nodes.

We now apply this result (on the existence of the subtree $\hat{T}$) to prove the lemma. Initialize T to be the original tree, with n nodes. Let $T_1 = \hat{T}$ be the subtree satisfying properties (a) and (b). Then, remove from T all non-root nodes of T_1; remove also the root, u_1, of T_1, if all descendents of u_1 in T are also nodes of T_1. Now, for the new tree T, let $T_2 = \hat{T}$ be the subtree satisfying properties (a) and (b). Continue in this way, defining $T_1, T_2, T_3, \ldots$ until the size of T falls below $\sqrt{n}$. Each subtree defined in this way has at most $\sqrt{n}$ nodes (and at least $\sqrt{n}/2$ nodes). Further, T_i can share at most one node (its root) with another subtree T_j, with $j > i$. $\square$

Define $t(T)$ to be a awakening time of tree T with the density-based algorithm. By counting time cost on each step, we have

$$t(T) \le t(T') + 4h + 2h + 0 + 2h + \max_{T_i} \{t(T_i)\}\,.$$

Now with the formula above we prove Theorem 2 by inductively proving the claim:

CLAIM 1. *For any unweighted tree T with height $h(T)$ and $N(T)$ nodes, and a sufficiently large constant C,*

$$t(T) \le [C(\log N(T))^{\log(5/3)} - 12] \cdot \max(h(T), \log N(T)).$$

PROOF. The proof is by induction. When $h(T) = 1$ or $N(T) \le 4$, the claim is trivial, if we choose a large enough C (21 suffices).

Now we make the induction hypothesis that for all trees T such that $N(T) < m$ ($m \ge 4$), the claim holds. Consider a tree T with $N(T) = m$. By the induction hypothesis applied to the dense tree T', the awakening time $t(T')$ of tree T' satisfies

$$\begin{aligned} t(T') &\le [C(\log N(T'))^{\log(5/3)} - 12] \cdot \\ &\qquad \max(h(T'), \log N(T')) \\ &\le [C(\log \sqrt{N(T)})^{\log(5/3)} - 12] \cdot \\ &\qquad \max(\left\lceil \frac{h(T)}{2} \right\rceil, \log \sqrt{N(T)}) \\ &\le (2/3)[C(\log \sqrt{N(T)})^{\log(5/3)} - 12] \cdot \\ &\qquad \max(h(T), \log N(T))\,. \end{aligned}$$

(The (2/3) arises in the last inequality, rather than a (1/2), in order to account for the rounding up in the $\lceil h(T)/2 \rceil$ expression in the case that $h(T) = 3$.) Then, applying the induction hypothesis to the subtrees T_i, the awakening time $t(T_i)$ satisfies

$$\begin{aligned} t(T_i) &\le [C(\log \sqrt{N(T)})^{\log(5/3)} - 12] \cdot \max(h(T), \log \sqrt{N(T)}) \\ &\le [C(\log \sqrt{N(T)})^{\log(5/3)} - 12] \cdot \max(h(T), \log N(T))\,. \end{aligned}$$

Thus, the awakening time $t(T)$ of tree T satisfies

$$\begin{aligned} t(T) &\le 8h(T) + (5/3)[C(\log \sqrt{N(T)})^{\log(5/3)} - 12] \cdot \\ &\qquad \max(h(T), \log N(T)) \\ &\le [8 + C(5/3)(\log \sqrt{N(T)})^{\log(5/3)} - 20] \cdot \\ &\qquad \max(h(T), \log N(T)) \\ &= [C(5/3)(0.5 \log N(T))^{\log(5/3)} - 12] \cdot \\ &\qquad \max(h(T), \log N(T)) \\ &= [C(\log N(T))^{\log(5/3)} - 12] \cdot \\ &\qquad \max(h(T), \log N(T))\,. \end{aligned}$$

$\square$

Combining this with Lemma 1, we have proved Theorem 2.

THEOREM 3. *The algorithm requires $O(n^2)$ time.*

PROOF. It takes $O(n)$ preprocessing time to calculate the number of children for all the nodes. Then for each recursive step, the extra time is $O(n^2)$ to find the dense subtree and determine how to split the subtrees. Thus for the processing time $t(n)$ we have the recursive formula $t(n) \le 2t(n/2) + O(n^2)$, which means the whole algorithm is $O(n^2)$ running time. $\square$

3. SIBLING-BASED ALGORITHM

In this section we present another approach to solve the FTP. As before we find a small-height spanning tree T. The main idea of our *sibling-based algorithm* is that each sibling in the tree T dumps the awakening work onto its "smaller" siblings.

3.1 Unweighted Graphs

In this section we prove the following theorem for the FTP on unweighted graphs.

THEOREM 4. *There is an $O(1)$-approximation algorithm for the FTP in unweighted graphs.*

Together with Lemma 1, we can conclude the following:

COROLLARY 1. *In an unweighted graph G with n nodes and diameter d the optimal makespan is $\Theta(\log n + d)$.*

The Algorithm. We first find a spanning tree T of graph G rooted at v_0, whose height is at most d (e.g., using breadth-first search), and we root the tree at v_0. Define h to be the height of tree T, which is between $d/2$ and d. As before the awakening algorithm will now only use the edges in T.

We define the *priority of node* v, $p(v)$, to be the number of nodes in the subtree rooted at node v. We denote nodes $v_1, v_2, \ldots, v_k$ to be the k children of root node v_0, and we assume they are sorted in decreasing order by priority.

The algorithm is as follows: The first awake robot at v_0 begins to awaken robots at children nodes $v_1, v_2, \ldots, v_k$ in order of decreasing priority; once it has awakened all of these children, the robot stops. When the robot at $v_i, i \geq 2$ is awakened, it rouses the robots at *two* sibling nodes in order of decreasing priority. Then it returns to node v_i and recursively awakens the subtree rooted at v_i. (The fact that the robot awakened at v_1 behaves differently from those at other siblings $v_i, i \geq 2$, is only necessary for the case of graphs with edge weights, in Section 3.3.)

Thus at every other time step, there is a group of robots at v_0. Some of these robots return to their original nodes to awaken the subtrees. The rest of the robots in the group rouse the sleeping robots with the highest priorities.

LEMMA 4. *The robot at node v_j will be awakened by time $4\lceil \log j \rceil + 1$.*

PROOF. At time step $i = 2$, there are two robots at v_0, one from v_0, another from v_1.

Suppose that at time step $2i$ we have m active robots at the root v_0. Then at most $1/4$ of these robots at v_0 are going to awaken their original subtrees. This is because whenever a robot stops awakening siblings, it has 3 replacement robots (two that it awakened personally, one that was awakened by the first sibling robot that is awakened). Therefore at least $\frac{3}{4}m$ robots at v_0 will awaken children of v_0 at time step $2i+1$. Consequently at time step $2i+2$ at least $\frac{3}{2}m$ active robots are at v_0.

So by induction, at time step $2i$ at least $(3/2)^i$ active robots are at v_0. Because $v_1, v_2, \ldots, v_j$ are awakened in order of decreasing priority, if there are $1+j$ active robots at time step i, v_j is awakened by that time. Therefore the robot at v_j is awakened by the smallest time step $2i$ such that $(3/2)^i \geq (1+j)$. This condition means the robot at node v_j will be awakened by time step $4\lceil \log j \rceil + 1$. $\square$

COROLLARY 2. *The recursive awakening of the subtree rooted at v_j starts by time $4\lceil \log j \rceil + 7$.*

PROOF. When the robot at node v_j is awakened, it awakens two siblings and returns to its original position v_j to start recursively awakening the robots in the subtree rooted at v_j. Therefore by Lemma 4 the corollary follows. $\square$

LEMMA 5. *For all $j = 1, 2, \ldots, k$, $p(v_j) \leq n/j$.*

PROOF. The lemma follows from $p(v_1) \geq p(v_2) \geq \cdots \geq p(v_j)$ and $p(v_1) + p(v_2) + \cdots + p(v_j) \leq n$. $\square$

LEMMA 6. *For an unweighted tree with n nodes and height h, the algorithm awakens all robots in time $\Theta(\log n + h)$.*

PROOF. We prove by induction on h that there exists a constant c, such that the makespan is at most $c(\log n + h)$:

For an unweighted tree with height $h = 1$, the algorithm has a makespan of $2\log n$, since the distance between any pair of leaves is at most 2.

For an unweighted tree with height h, suppose by induction that each subtree rooted at v_j for $j = 1, 2, \ldots, k$ can be awakened by time $c[\log(p(v_j)) + h - 1]$. As long as $c \geq 11$, it follows from Corollary 2 and Lemma 5 that all the nodes in the subtree are awakened by time

$$\begin{aligned} t &= 4\lceil \log j \rceil + 7 + c(\log(p(v_j)) + h - 1) \\ &\leq 4\log j + 11 + c\log(n/j) + ch - c \\ &< c\log n + ch. \end{aligned}$$

Thus we proved the inductive steps for the trees of height h establishing the upper bound. It follows from Lemma 1 that this bound is tight within a constant factor. $\square$

This concludes the proof of Theorem 4.

THEOREM 5. *The Sibling-Based Algorithm requires linear time.*

PROOF. Using a breadth-first search algorithm, we can find a spanning tree in linear time, and it takes linear parallel running time to assign priority to each node (counting the number of children for the node). The total time for a robot to make decisions is linear in the sum of the node degrees which the robot needs to walk through during the whole awakening process, which is $O(n)$. Thus the total running time is linear. $\square$

NP-Hardness of FTP on Unweighted Graphs.

THEOREM 6. *The Freeze-Tag Problem is NP-Hard for unweighted graphs (i.e. all edge lengths are one) with exactly one robot at each node.*

PROOF. The proof is a variant of the one presented in [5], which shows that FTP on general weighted graphs is NP-hard to approximate within a factor better than 5/3.

The reduction is from 3SAT. Without loss of generality, we assume that n, the number of variables, is even. For technical reasons, we include n clauses of size 2 of the form "x or $\bar{x}$", one for each variable. We let c be the maximum number of clauses in which a literal appears. We now describe the construction of the graph. Let r be a root node, the node containing the only "awake" robot. From r we have a path to a node p, which includes $n/2$ nodes (including r and p). Clearly, the path has $p/2 - 1$ edges. Next, there are $p/2$ preliminary nodes $p_1, \ldots p_{n/2}$, with an edge (p, p_i) for $1 \leq i \leq p/2$.

Next, we group the n variables in an arbitrary way to $n/2$ pairs, and assign a pair to each preliminary node. Each pair of variables, say j, k is represented by four vertices, x_j, $\bar{x}_j$, x_k, $\bar{x}_k$, with edges to these four vertices from the preliminary node p_i assigned to them. In addition, we have an edge $(x_j, \bar{x}_j)$ for every variable j.

From x_j (resp. $\bar{x}_j$) we have a path containing c nodes ($c-1$ edges) to a literal node y_j (resp. $\bar{y}_j$).

Finally, there is a node for each clause (including the clauses of size 2 we included) with an edge from literal node y_j (or $\bar{y}_j$) to each clause in which it appears.

Recall that all edges have unit length.

We claim that all robots can be awakened in time $n/2 - 1 + 1 + 1 + c - 1 + 1 = n/2 + c + 1$ if and only if the formula is satisfiable. A satisfying truth assignment can be turned into such a wake-up tree, easily: The robot at r travels to p waking all robots along the way, arriving at

p at time $n/2-1$. There are now $n/2$ awake robots at p. Each such robot travels to a different preliminary node p_i (another 1 time unit), and now there are 2 awake robots at each preliminary node. Each such awake robot goes to a true literal (1 more time unit). Now, at this true literal, say x_j, there are 2 awake robots. One goes down the path to the y_j node, (there are now c awake robots at y_j) and then to all clauses the literal appears in, which takes time $c-1+1$. The other robot goes to the false literal, $\bar{x}_j$ then down its path to the $\bar{y}_j$ node, which also take time $1+c-1$. Since each clause has a true literal, all robots are awake, and the process took $n/2-1+1+1+c$.

Conversely, a solution of makespan $n/2+c+1$ induces a satisfying truth assignment, as the only way to wake up all robots at clause nodes in this time is to have a wake up path to them from r to p to p_i to a variable node x, down the path to a literal node y to the clause node. Any other path must be longer.

Note that if there is no satisfying truth assignment, then the makespan is longer by only 1 time unit; thus, our reduction does not give a hardness of approximation result.

Note also that we can easily modify the graph to have node degrees of at most 5, by replacing the paths with trees where internal nodes have 2-3 children. $\square$

3.2 Unweighted Edges but Weighted Nodes

We consider now a graph G having unweighted (without loss of generality, unit-length) edges, but having weighted nodes, i.e., there may be multiple asleep robots initially at a node. When there are multiple asleep robots at a node, $\log n$ is no longer a lower bound on the makespan of an awakening strategy; $d/2$ is still a lower bound because the signal still needs to travel to the farthest node.

We show how a simple variant of the sibling-based algorithm results in an $o(\log n)$-approximation algorithm for this case of the FTP; this improves the $O(\log n)$-approximation bound that follows from the general results of [5].

THEOREM 7. *There exists an $O(\sqrt{\log n})$-approximation algorithm for the FTP on graphs with unweighted edges but weighted nodes.*

PROOF. We propose the following *Hybrid Algorithm*:

Hybrid Algorithm.

1. If the diameter $d \geq \sqrt{\log n}$, then we apply the sibling-based algorithm from Section 3.1, only using *one* of the robots originally placed at each node during the awakening process.

2. If the diameter $d < \sqrt{\log n}$, then we activate nodes in G in the order of decreasing number of robots on the nodes. (We completely ignore the topology of the tree.)

In Case 1, the entire tree is awakened in time $O(d+\log n)$. Since $d/2$ is a lower bound on OPT and $d \geq \sqrt{\log n}$, we see that $d+\log n = O(d\sqrt{\log n})$, so the algorithm is an $O(\sqrt{\log n})$-approximation.

In Case 2, we consider the complete graph $G' = K_n$ on the n nodes of G, with all edges of G' having unit length. An optimal awakening strategy for G' is to awaken the nodes in order of decreasing node weight (number of asleep robots at the node); the fact that this is optimal follows easily by an exchange argument, as in [5]. The time required to awaken all robots in G' is a lower bound on the optimal time to awaken all robots in G, since the distance in G between any two nodes is at least the distance (one) between the nodes in G'. Since the maximum distance between two nodes in G is at most $d \leq \sqrt{\log n}$, this algorithm awakens G in time at most d times the time to awaken G'. Thus, we obtain an $O(\sqrt{\log n})$-approximation. $\square$

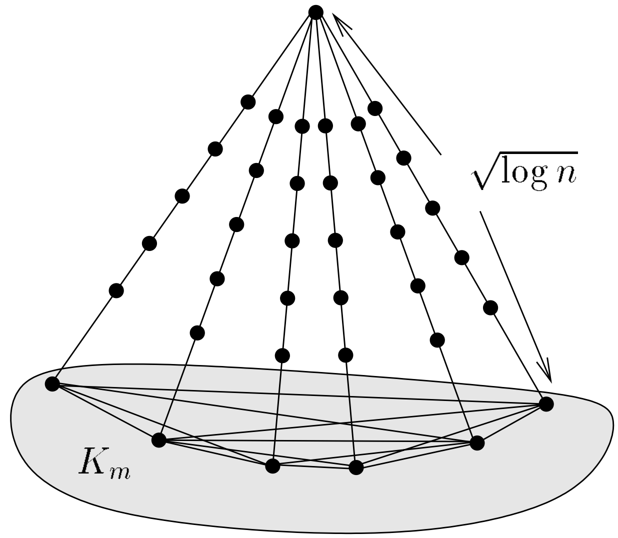

Figure 1: Example of a "bad" spanning tree in a graph G with unweighted edges. Here, each chain is of length $\sqrt{\log n}$, the $\Theta(n/\sqrt{\log n})$ bottom level nodes form a complete graph, each edge has unit length, each bottom level node has initially exactly $2^{\sqrt{\log n}}$ asleep robots, and all other nodes have initially exactly one asleep robot.

Limitation of the Above Method. The algorithms in this paper begin by finding an arbitrary approximate minimum-height spanning tree T, where the edges in the spanning tree are the only edges that the robots ever traverse. Unless the algorithms are more careful in choosing T, we cannot hope to obtain an approximation factor better than $O(\sqrt{\log n})$ for the the case of unweighted edges and weighted nodes, as we now show:

THEOREM 8. *Consider the FTP on a graph G having unweighted edges and weighted nodes; any algorithm based on an (arbitrary) approximate-minimum-height spanning tree is an $\Omega(\sqrt{\log n})$-approximation algorithm.*

PROOF. Consider the example shown in Figure 1. The unique minimum-height spanning tree T of graph G is a star of chains of length $\sqrt{\log n}$, with each of the $\Theta(n/\sqrt{\log n})$ leaf nodes having exactly $2^{\sqrt{\log n}}$ asleep robots and each of the $\Theta(n)$ non-leaf nodes having exactly one robot. In G, any two leaf nodes are at distance 1, while in T they are at distance $2\sqrt{\log n}$ from each other.

The optimum awakening time for the tree T is $\Theta(\log n)$; we can achieve $O(\log n)$ time, e.g., using the sibling-based algorithm. The $2^{\sqrt{\log n}}$ robots at each leaf cannot assist in awakening any other robots for at least $\sqrt{\log n}$ steps, implying that they cannot contribute to a faster awakening algorithm than $\Theta(\log n)$. In contrast, an awakening in G can be achieved in time $\Theta(\sqrt{\log n})$ by first sending one robot to awaken one leaf in T, then awakening all of the leaves in time $\Theta(\sqrt{\log n})$, and then awakening the remaining nodes (along the chains) in another $\Theta(\sqrt{\log n})$ steps. $\square$

From Theorem 8, we immediately obtain the following corollary, showing that our analysis of the Hybrid Algorithm is tight:

THEOREM 9. *The Hybrid Algorithm is a $\Theta(\sqrt{\log n})$-approximation algorithm for the FTP on graphs with unweighted edges but weighted nodes.*

3.3 General Graphs

We continue to assume, without loss of generality, that the shortest edge of G has length 1, but now we allow longer edges as well. We let L denote the length of the longest edge and let d denote the diameter of G.

As before, we find a spanning tree T for the graph G, where the height h of T is between $d/2$ and d, and we run the sibling-based algorithm on the spanning tree T.

Using notation and arguments similar to the unweighted case, we prove the following lemmas and corollaries. First, note that Lemma 5 still holds for the weighted case. Lemma 4, Corollary 2, and Lemma 6 become the following:

LEMMA 7. *The robots at node v_j are awakened by time $4L\lceil\log j\rceil + d(v_0, v_j)$.*

PROOF. The proof of Lemma 4 implies that by time $4L\lceil\log j\rceil$ there is a robot that has reached the root node v_0 and has at least begun its journey down edge (v_0, v_j) to awaken the robots at v_j. This implies that the asleep robots at node v_j will be awakened by time $4L\lceil\log j\rceil + d(v_0, v_j)$. $\square$

COROLLARY 3. *The recursive awakening process of the subtree rooted at v_j starts by time $4L\lceil\log j\rceil + d(v_0, v_j) + 6L$.*

PROOF. When the robot at node v_j ($j \geq 2$) is awakened, it awakens two siblings and returns to its original position v_j to start recursively awakening the robots in the subtree rooted at v_j. The travel to awaken the two siblings and then return to v_j involves 6 edge traversals, each of length at most L; thus, by Lemma 7, the claim follows. $\square$

LEMMA 8. *For the tree T, the sibling-based algorithm awakens all of the robots by time $O(h + L\log n)$.*

PROOF. We prove, by induction on the number of levels of tree T, that there exists a constant, c, such that the makespan is at most $c\,(L\log n + h)$.

For a weighted tree with one level (i.e., a weighted star), the algorithm has a makespan of at most $2L\log n$, since the distance between any pair of leaves is at most $2L$.

For a weighted tree T rooted at v_0, suppose by induction that each subtree rooted at the children of v_0, v_j for $j = 1, 2, \ldots, k$, can be awakened by time $c\,[L\log(p(v_j)) + h - d(v_0, v_j)]$. As long as $c \geq 11$, Corollary 3 and Lemma 5 imply that all nodes in the subtree are awakened by time

$$\begin{aligned} t &= 4L\lceil\log j\rceil + 6L + d(v_0, v_j) + c[L\log(p(v_j)) + h - d(v_0, v_j)] \\ &\leq 4L\log j + 10L + cL\log(n/j) + ch - (c-1)d(v_0, v_j) \\ &< cL\log n + ch \end{aligned}$$

if $j \geq 2$.

If $j = 1$, then, since the recursive awakening process of the subtree rooted at v_j starts at time $d(v_0, v_j)$, we know that the subtree is awakened by time

$$\begin{aligned} t &= d(v_0, v_1) + c[L\log(p(v_j)) + h - d(v_0, v_1)] \\ &\leq c(L\log(p(v_j)) + h) \\ &\leq c(L\log n + h)\,, \end{aligned}$$

which completes the proof, by induction. (We remark that the distinction between the $j \geq 2$ and $j = 1$ cases is necessary because the subtree rooted at the first child, v_1, may contain almost all of the descendents of v_0, while, in contrast, we can bound the number of descendents in the subtrees of v_j for $j \geq 2$.) $\square$

Note that the only difference in the proof of Lemma 8, compared with Lemma 6, is that we prove, by induction, that there exists a constant c such that the makespan is $c(h + L\log n)$ instead of $c(h + \log n)$. The proofs for Lemma 7 and Corollary 3 follow similarly to the unweighted case, except that the edge lengths are different and are bounded by L or $d(v_j, v_0)$. The important observation of Lemma 8 is that the L multiplies the $\log n$ term but does not multiply the height h. We obtain the following theorem:

THEOREM 10. *For a weighted graph G with longest edge length L, diameter d, and n nodes, the algorithm awakens all of the robots by time $O(d + L\log n)$.*

The following corollary isolates the difficult case for the FTP in general graphs.

COROLLARY 4. *For a graph G with n nodes, diameter d, and longest edge length L, the sibling-based algorithm gives an $O((L/d)\log n)$-approximation algorithm for the FTP problem. Thus, if $L = o(d)$, the sibling-based algorithm is an $o(\log n)$-approximation algorithm.*

4. BICRITERIA VERSION OF FTP

In this section we consider a *bicriteria* optimization problem for the FTP. Specifically, we simultaneously optimize the *makespan* and *the total distance travelled* by all of the robots in the awakening process. We use the following notation: If the makespan of an algorithm is at most $f(n)$ times the optimal makespan *and* the total distance travelled by all robots is at most $g(n)$ times the optimal total travel distance, then we say that the algorithm is an $\langle\, f(n),\, g(n)\,\rangle$-approximation.

We prove the following theorem for the bicriteria version of the FTP:

THEOREM 11.
1. *For an* unweighted *graph G, there exists an $\langle\, O(1),\, O(1)\,\rangle$-approximation algorithm for the FTP.*
2. *For a graph G with* weighted edges and unweighted nodes, *there exists an $\langle\, O(\sqrt{\log n}),\, O(\sqrt{\log n})\,\rangle$-approximation algorithm for FTP.*
3. *For any* general *graph G, there exists an $\langle\, O(\log n),\, O(1)\,\rangle$-approximation algorithm for the FTP.*

The proof of Theorem 11 is based on the next two theorems. We define the weight $W(T)$ of a tree T to be the sum of the edge lengths in the tree T.

THEOREM 12 ([6]). *For any graph G and any $\varepsilon > 0$, there exists a spanning tree T of G whose weight is at most $2(1 + O(\epsilon))$ times the weight of a minimum spanning tree of G, and whose diameter is at most $1 + O(1/\epsilon)$ times the diameter of G.*

Next we show that the minimum total distance travelled in a tree is order of the weight of the tree.

THEOREM 13. *For any tree T, the sibling based algorithm is a 4-approximation for the problem of minimizing the total distance travelled on T.*

PROOF. Since the awakening signal must travel through all the edges to awaken all of the robots, OPT is at least $W(T)$. With the sibling-based algorithm, each edge is traversed at most three times; it follows that the total distance travelled with the sibling-based algorithm is at most $4W(T)$. □

Proof of Theorem 11:

First, we select the spanning tree T according to Theorem 12. Then we use the sibling-based algorithm on the tree T. According to Corollary 1, we know it is an $O(1)$-approximation for the makespan. It follows from Theorem 13 and Theorem 12 that this algorithm gives an $O(1)$-approximation for the total distance travelled.

If the diameter d is greater than $\sqrt{\log n}$, we select the spanning tree T as above, and apply the sibling-based algorithm to it. According to Corollary 1 it is an $O(\sqrt{\log n})$-approximation for the makespan. It follows from Theorem 13 and Theorem 12 that this algorithm gives an $O(1)$-approximation for the total distance travelled.

If the diameter d is less than $\sqrt{\log n}$, we use the greedy algorithm, ignoring the graph structure. It follows from Theorem 7 that it is an $O(\sqrt{\log n})$-approximation for the makespan. Since a spanning tree has $n - 1$ edges, the total distance travelled by all the robots is at least $n - 1$. Thus, the greedy algorithm gives an $O(1)$-approximation for the total distance travelled.

For a general graph G, we select the spanning tree T as above and use the sibling-based algorithm on it. It follows from Theorem 10 that it is an $O(\log n)$-approximation for the makespan. It follows from Theorem 13 and Theorem 12 that this algorithm gives an $O(1)$-approximation for the total distance travelled. □

5. ACKNOWLEDGMENTS

E. Arkin and J. Mitchell are partially supported by NSF (CCR-0098172). M. Bender is partially supported by NSF (EIA-0112849, CCR-0208670) and by a grant from Sandia National Labs. J. Mitchell acknowledges support from Honda Research Labs, NASA Ames Research (NAG2-1325), and the U.S.-Israel Binational Science Foundation.

6. REFERENCES

[1] S. Albers and M. R. Henzinger. Exploring unknown environments. In *Proc. 29th ACM Sympos. Theory Comput.*, pages 416–425, 1997.

[2] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 842–843, 1999.

[3] S. Alpern and S. Gal. *Search Games and Rendezvous Theory.* Kluwer Academic Publishing, to appear.

[4] E. J. Anderson and S. P. Fekete. Two-dimensional rendezvous search. *Operations Research*, 49:107–118, 2001.

[5] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. B. Mitchell, and M. Skutella. The Freeze-Tag Problem: How to wake up a swarm of robots. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 568–577, 2002.

[6] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proc. 9th ACM Sympos. on Principles of Distributed Computing*, pages 177–187, 1990.

[7] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *Proc. 30th ACM Sympos. Theory Comput.*, pages 448–453, 1998.

[8] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proc. 30th ACM Sympos. Theory of Comput.*, pages 269–278, 1998.

[9] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176:1–21, 2002.

[10] M. A. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proc. 35th IEEE Sympos. Found. Computer Science*, pages 75–85, 1994.

[11] A. M. Bruckstein, C. L. Mallows, and I. A. Wagner. Probabilistic pursuits on the grid. *American Mathematical Monthly*, 104(4):323–343, 1997.

[12] A. Dumitrescu, I. Suzuki, and M. Yamashita. High speed formations of reconfigurable modular robotic systems. In *Proc. IEEE Internat. Conference on Robotics and Automation*, pages 123–128, 2002.

[13] D. Gage. Many-robot systems. URL: `http://www.spawar.navy.mil/robots/research/manyrobo/manyrobo.html`.

[14] D. Gage. Sensor abstractions to support many-robot systems. In *Proc. SPIE Mobile Robots VII*, Boston MA, Vol. 1831, pages 235–246, 1992.

[15] D. Gage. An evolutionary strategy for achieving autonomous navigation. In *Proc. SPIE: Mobile Robots XIII*, Boston MA, Vol. 3525, 1998.

[16] D. Gage. Minimum-resource distributed navigation and mapping. In *Proc. SPIE Mobile Robots XV*, Vol. 4195, 2000.

[17] S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *NETWORKS*, 18:319–349, 1988.

[18] T.-R. Hsiang, E. Arkin, M. A. Bender, S. Fekete, J. Mitchell Algorithms for rapidly dispersing robot swarms in unknown environments. In *Proc. 5th Workshop on Algorithmic Foundations of Robotics*, 2002.

[19] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring an unknown cellular environment. In *Abstracts 16th European Workshop Comput. Geom.*, pages 140–143, 2000.

[20] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proc. 35th Sympos. Found. Computer Science*, pages 202–213, 1994.

[21] N. Roy and G. Dudek. Collaborative exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[22] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`.

[23] R. V. Solé, E. Bonabeau, J. Delgado, P. Fernández, and J. Marín. Pattern formation and optimization in army ant raids. *Artificial Life*, 6(3):219–226.

[24] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems*, 13(3):127–139, 1996.

[25] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[26] M. Sztainberg, E. M. Arkin, M. A. Bender, and J. S. B. Mitchell. Analysis of heuristics for the Freeze-Tag Problem. In *Proc. Scandinavian Workshop on Algorithms*, Vol. 2368 of *Springer-Verlag LNCS*, pages 270–279, 2002.

[27] I. Wagner and A. Bruckstein. Cooperative cleaners: a study in ant robotics. Technical Report CIS9512, Technion, 1995.

[28] I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Trans. Rob. Aut.*, 15(5):918–933, 1996.

[29] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Efficiently searching a graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):211–223, 1998.

[30] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. MAC vs. PC – Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *Internat. Journal Robotics Research*, 19(1): 12–31, 2000.