# Analysis of Heuristics for the Freeze-Tag Problem

Marcelo O. Sztainberg  Esther M. Arkin  Michael A. Bender

Joseph S. B. Mitchell

April 20, 2002

**Abstract**

In the Freeze Tag Problem (FTP) we are given a swarm of $n$ asleep (*frozen* or *inactive*) robots and a single awake (*active*) robot, and we want to awaken all robots in the shortest possible time. A robot is awakened when an active robot "touches" it. The goal is to compute an optimal *awakening schedule* such that all robots are awake by time $t^*$, for the smallest possible value of $t^*$. We devise and test a variety of heuristic strategies on geometric and network datasets. Our experiments show that all of the strategies perform well, with the simple greedy strategy performing particularly well. A theoretical analysis of the greedy strategy gives a tight approximation bound of $\Theta(\sqrt{\log n})$ for points in the plane. We show more generally that the (tight) performance bound is $\Theta((\log n)^{1-1/d})$ in $d$ dimensions. This is in contrast with the case of general metric spaces, where greedy is known to have a $\Theta(\log n)$ approximation factor, and no method is known to achieve an approximation bound of $o(\log n)$.

## 1 Introduction

We consider a natural problem that arises in the study of *swarm robotics*. Consider a set of $n$ *robots*, modeled as points in some metric space. There is one "awake" *source* robot; all other robots are *asleep* (inactive). In order to awaken a sleeping robot, an active robot travels to it and touches it; then, that robot can assist the set of active robots in awakening other asleep robots. Our goal is to activate (wake up) all of the robots as quickly as possible; i.e., we want to minimize the *makespan*, which is the time when the last robot is awakened.

This problem has been coined the *freeze-tag problem* (FTP) [1] because of its similarity with the children's game of "freeze-tag." In the game, the person who is "it" tags a player, who becomes "frozen" until another player (who is not "it" and not "frozen") tags him to unfreeze him. The FTP arises when there are a large number of players that are frozen, and one (not "it") unfrozen player, whose goal it is to unfreeze the rest of the players as quickly as possible. Once a player gets unfrozen, s/he is available to assist in unfreezing other frozen players, who can then assist, etc. Other applications of the FTP arise in the context of distributing data (or some other commodity), where physical proximity is required for distribution. This proximity may be necessary because wireless communication has too high a bandwidth cost or security risk. How does one propagate the data to the entire set of participants in the most efficient manner? Prior work on the dissemination of data in a graph includes the *minimum broadcast time problem*, the *multicast problem*, and the related *minimum gossip time problem*; see [4] for a survey and [2, 6] for recent approximation results.

The FTP is expressed as a combinatorial optimization problem as follows: Given a set of points in a metric space, find an arboresence (*awakening tree*) of minimum height where every node has out-degree at most two.

What makes the freeze-tag problem particularly intriguing is that *any* reasonable ("nonlazy") strategy yields an $O(\log n)$-approximation (Proposition 1.1 of [1]), whereas no strategy is known for general metric spaces that yields a $o(\log n)$-approximation. Arkin et al. [1] show that even very simple versions of the problem (e.g., on 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 $d$. They also give a variety of results on star metrics, where $O(1)$-approximation is possible, and an $o(\log n)$-approximation for the special case of *ultrametrics*.

In this paper, our main results include the following:

**(1)** A proof that the natural greedy heuristic applied to geometric instances gives an $O((\log n)^{1-1/d})$-approximation in $d$ dimensions. Thus, in one dimension, the greedy heuristic yields an $O(1)$-approximation, and in the plane the greedy heuristic yields an $O(\sqrt{\log n})$-approximation. We prove that this analysis is tight by supplying matching lower bounds.

While better approximations are known for the geometric instances (indeed, there is an efficient PTAS for any constant $d$ dimensions [1]), the greedy strategy is not only much simpler and more natural, but also it is implementable *on-line*, in which each robot selects the next robot it will awaken at the instant that it wakes up, based on which asleep robot is closest to it.

**(2)** We perform an experimental investigation of heuristic strategies for the FTP, comparing the different choices of greedy strategies and comparing these greedy strategies with other heuristics. We present experimental results on classes of randomly generated data, as well as on datasets from the TSPLIB repository.

**Notation.** We let $S$ denote the *swarm*, the set of $n$ points in a metric space (often Euclidean $d$-space, denoted $\Re^d$) where the initially asleep robots are located. We let $s$ denote the *source point* where an initially active source robot is placed. We assume that any active robot in motion travels with unit speed. We let $R$ denote the *radius* of the swarm $S$ with respect to $s$; i.e., $R = \max_{p \in S} \mathrm{dist}(s,p)$. We let $D = \max_{p,q \in S \cup \{s\}} \mathrm{dist}(p,q)$ denote the diameter of the set of robots. We let $t^*$ denote the minimum makespan. Note that, trivially, $t^* \geq R$ (since robots move with unit speed).

## 2 Wakeup Strategies for the FTP

We begin by describing the class of greedy awakening strategies, which we analyze, giving lower and upper bounds on their performance in geometric datasets. At the end of this section, we describe other strategies that we have devised and tested in our experiments.

### 2.1 Greedy Strategies

A natural strategy for the FTP is the greedy strategy, where an awake robot chooses the nearest asleep robot to awaken. The motivation for awakening nearby robots first is that parallelism is generated early on: the first robot awakens its closest neighbors, these newly awakened robots awaken their closest sleeping neighbors; thus there is rapid exponential growth in the number of awake robots.

In fact the greedy strategy is not a fully defined heuristic. What remains to be specified is how conflicts among robots are resolved, since two robots may have the same closest neighbor. We now describe three methods for resolving these conflicts: claims, refresh, and delayed target choice.

**Claims.** In order to guarantee that work is not duplicated, an awake robot *claims* the sleeping robot that it intends to awaken. Once a sleeping robot is claimed, no other robot is allowed to claim or awaken it.

**Refresh.** It may be beneficial for newly awakened robots to "renegotiate" the claims, since the set of awake robots changes. We refer to the ability to reassign active robots to asleep robots as the option to *refresh* claims.

We first consider the case in which claims are *not* adjusted. Thus, once an awake robot $A$ claims an asleep robot $B$, $A$ awakens $B$ before making any other algorithmic decisions. The algorithm is now well defined because, without loss of generality, at most one robot is awakened at a time, and each time a robot is awakened it claims the nearest asleep robot.

Consider now the case in which claims *are* renegotiated, or *refreshed*, each time a new robot awakens. For motivation, consider the following scenario. An awake robot $A$ is heading toward a sleeping robot $B$, which $A$ has claimed. Before $A$ reaches $B$, another robot $C$ awakens. Now, both $A$ and $C$ would like to claim $B$, but since $B$ is closer to $C$ than to $A$, $C$ takes over the responsibility of awakening $B$.

In our experiments, we assign claims by finding a matching between the awake robots and the asleep robots. We use a (potentially suboptimal) greedy strategy to compute a matching, rather than applying a more complex optimization algorithm. We order, by length, the potential matching edges between the set of currently awake and currently sleeping robots. We iteratively add the shortest edge $e$ to the matching, and remove from consideration those edges that are incident to either of $e$'s endpoints. While the resulting matching need not have minimum total weight, it has the property of giving priority to the short matching edges, which is faithful to the greedy heuristic.

**Delayed Target Choice.** Refresh introduces several anomalies. In particular, an (awake) robot may repeatedly change directions and oscillate without awakening any robots. This happens when an (awake) robot chooses an asleep target, but before the robot reaches its target, another robot claims the target. With the *delayed target choice* option we avoid these oscillations by making the solution "more off-line". Specifically, we avoid committing to the direction a robot is heading until that robot has traveled far enough to awaken its target, at which point the target (and its position) is fully determined. Thus, the greedy strategy proceeds as follows. For each robot we store the last position, $p$, of the previous robot that it awakened. We maintain the maximum distance (i.e., the time), $x$, that it traveled since the last time it awakened a robot. When considering candidate matching edges to link $p$ to the robot's next target $q$, we subtract $x$ from the actual distance $\mathrm{dist}(p, q)$, because the robot has already traveled this distance.

### 2.1.1 Lower Bounds on the Performance of the Greedy Heuristic

We now present lower bounds on the performance of the greedy heuristic on sets of points in $d$-dimensional Euclidean space. We begin with $d = 1, 2$, and then consider the general case of higher dimensions. Our lower bounds hold for all variations of the greedy heuristic described above (with or without claims, refresh, or delayed target choice), since, in our lower bound examples, all awake robots will travel together in a "pack" so that all awake robots make claims simultaneously.

**Theorem 1** *For any $\epsilon > 0$, there exists an instance of the FTP for points $S \subset \Re^1$ on a line ($d = 1$) for which the greedy heuristic results in a makespan that is at least $4 - \epsilon$ times optimal.*

**Proof:** We construct a family of instances in which the application of the greedy heuristic will result in all awake robots always staying together in a single group.

We place asleep robots at points $S = \{p_0, p_1, \ldots, p_n, p_{n+1}, p_{n+2}\}$, where $n = 2\ell$ is even. The source robot is placed at the origin, $s = 0$. We place one sleeping robot at point $p_0 = 1$, two sleeping robots at point $p_1 = 1 - 2 = -1$, four sleeping robots at point $p_2 = 1 - 2 + 4 = 3$, etc. In general, we place $2^j$ sleeping robots at point $p_j = \sum_{i=0}^{j}(-2)^i$ for $j = 1, 2, \ldots, n$. Note that $p_{2k} = \frac{2}{3}2^{2k} + \frac{1}{3}$ and $p_{2k+1} = -\frac{2}{3}2^{2k+1} + \frac{1}{3}$, so $p_{n-1} = -\frac{1}{3}2^n + \frac{1}{3}$ and $p_n = \frac{2}{3}2^n + \frac{1}{3}$. Finally, we place $2^{n+2}$ sleeping robots at point $p_{n+2} = p_{n-1} - 2^n = -\frac{4}{3}2^n + \frac{1}{3}$ and we place $2^{n+1}$ sleeping robots at $p_{n+1} = -p_{n+2} = \frac{4}{3}2^n - \frac{1}{3}$. Refer to Figure 1.

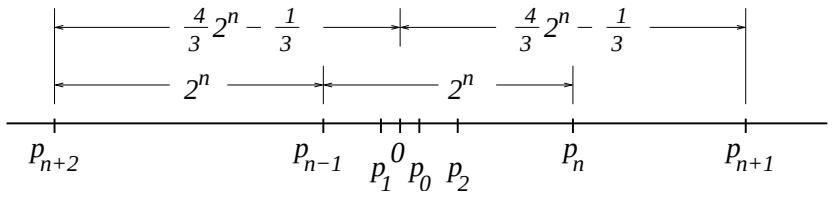


Figure 1: The lower bound construction in $\Re^1$, for which greedy is at least $4 - \epsilon$ times optimal.

The greedy strategy sends the source robot to the right, a distance of 1, to awaken the one robot at $p_0$, then two robots to the left, a distance of 2, to awaken the two robots at $p_1$, then four robots to the right, a distance of 4, to awaken the four robots at $p_2$, etc. (Actually, there are "ties" in making these decisions; however, the instance can be perturbed in order to make the decisions unique, using an infinitesimally small $\delta > 0$, placing the robots at points $p_{2k} = \frac{2}{3}2^{2k} + \frac{1}{3} - \delta^{2k+1}$ and $p_{2k+1} = -\frac{2}{3}2^{2k+1} + \frac{1}{3} + \delta^{2\ell+2}$.) The total distance traveled by the source robot up to the time it reaches $p_n$ is therefore $1 + 2 + 4 + 8 + \cdots + 2^n = 2 \cdot 2^n - 1$. Once the $2^n$ robots at $p_n$ have been awakened, the set of all $2^{n+1}$ awake robots goes to the *right*, a distance of $\frac{2}{3}2^n - \frac{2}{3}$, to awaken the robots at $p_{n+1}$, and then finally, all of the robots go to the *left* a distance of $\frac{8}{3}2^n - \frac{2}{3}$ to $p_{n+2}$. The makespan is given by the total distance traveled by the source robot, which is $(2 \cdot 2^n - 1) + (\frac{2}{3}2^n - \frac{2}{3}) + (\frac{8}{3}2^n - \frac{2}{3}) = \frac{16}{3}2^n - \frac{7}{3}$.

In contrast, an optimal awakening strategy is as follows. The source robot goes to the right, awakening one robot at $p_0$; then, the source robot continues heading to the right, awakening all of the robots at points $p_0, p_2, p_4, \ldots, p_{n-2}, p_n, p_{n+1}$, while the robot that was awakened at $p_0$ heads to the *left* to awaken all other robots $(p_1, p_3, \ldots, p_{n-1}, p_{n+2})$. The makespan is $1 + \frac{4}{3}2^n - \frac{1}{3} = \frac{4}{3}2^n + \frac{2}{3}$.

The ratio of the makespan of greedy to the optimal makespan approaches 4 (from below) as $n \to \infty$.

□

**Theorem 2** *The greedy heuristic is an $\Omega(\sqrt{\log n})$-approximation to the FTP in the plane.*

**Proof:** We arrange $\log n$ disks, each of radius 1, along a "zig-zag" path having $\sqrt{\log n}$ rows each having $\sqrt{\log n}$ disks, with disks touching but not overlapping along the path. Refer to Figure 2. The source robot is at the center, $s$, of the first disk. There is one asleep robot at the center of the second disk, then two at the center of the next, then four, etc, with the number of asleep robots at the center of each disk doubling as we advance along the path of touching disks. The last disk has (about) $\frac{n}{2}$ robots. (More precisely, we consider values of $n$ of the form $n = 2^{m^2-1} - 1$, for integer $m$. Then, the zig-zag has $m$ rows, each having $m$ disks. The last disk has $2^{m^2-2}$ robots at its center.) When the greedy strategy is applied to this example, the awakening happens in

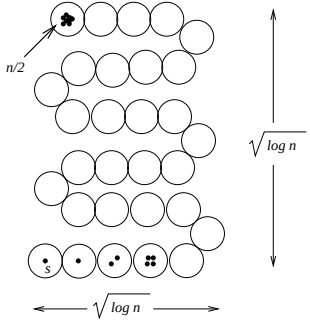


Figure 2: The lower bound construction in $\Re^2$.

sequence along the zig-zag path, with all of the newly awakened robots at the center of one disk targeting the robots at the center of the next disk. (There are just enough robots to allow a perfect matching, because of the doubling.) Thus, the greedy strategy takes $\log n$ steps, each of size about 2. An optimal strategy, however, sends the source robot vertically upwards, awakening one cluster of robots at the center of each disk it passes along the way. These robots are then available to travel horizontally, awakening the robots along each row. This strategy yields makespan $t^* = O(\sqrt{\log n})$. Hence, greedy is an $\Omega(\sqrt{\log n})$-approximation. □

Theorem 2 generalizes to $d$ dimensions.

**Theorem 3** *The greedy strategy is an $\Omega((\log n)^{1-1/d})$-approximation to the FTP in $\Re^d$.*

**Proof:** We arrange unit-radius disks along a zigzag path in $d$ dimensions; the construction is defined inductively by dimension, with $(\log n)^{1/d}$ copies of a $(d-1)$-dimensional construction, each within a layer, concatenated to form a path of touching (but not overlapping) disks in $\Re^d$. The numbers of asleep robots at the centers of the disks are $1, 2, 4, 8, \ldots$, and the one source robot is at the center of the first disk. As in the 2-dimensional case, the greedy strategy awakens robots in order along the path, taking a total of $\log n$ steps, each of size about 2. A much better strategy, though, is for the source robot to go to the center of the last disk (i.e., the $(\log n)^{th}$ disk) along the path, where $n/2$ robots can be awakened, who can then, in $O((\log n)^{1/d})$ time, spread out and awaken the other half of the robots. This strategy takes time $(\log n)^{1/d}$, implying that the greedy strategy is an $\Omega((\log n)^{1-1/d})$-approximation to the optimal strategy.

□

### 2.1.2 Upper Bounds for the Greedy Heuristic

We now present upper bounds for the greedy heuristic in $\Re^d$, for $d = 1, 2, \ldots$. We first show that the lower bound of Theorem 1 is essentially tight for the one-dimensional case.

**Theorem 4** *The greedy heuristic for a swarm $S \subset \Re$ of points on a line ($d = 1$) is a 4-approximation.*

**Proof:** Consider a longest path in the awakening tree. With each step from $p_i$ to $p_{i+1}$ of length $r_i$ along the path, we associate an interval of length $r_i$ of the line that is "charged" with the step: If the step is in the same direction (left or right) as the previous step, we charge the interval $(p_i, p_{i+1})$ of the step itself; we may assume that there are no asleep robots in this interval. If the step is in the opposite direction of the previous step, we charge the interval of length $r_i$ that touches $p_i$ but is on the other side of $p_i$ from $p_{i+1}$; by the greedy property, we know that this interval contains no asleep robots. It is easy to see that the intervals we charge are non-overlapping. The maximum length of all charging is at most $2D$, where $D$ is the diameter of the point set. Since the radius

$R \leq D/2$ is a lower bound on the optimal makespan, and the length of the longest path is at most $2D \leq 4R$, the claim follows.

□

We turn now to our analysis of the greedy strategy in two or more dimensions. Our goal is to prove the following theorems, which show that, in contrast with arbitrary metric spaces (where greedy gives an $O(\log n)$-approximation for the FTP), greedy is an $o(\log n)$-approximation for geometric instances. Combined with our lower bounds, we get a *tight* analysis of the approximation factor for greedy in all dimensions $d \geq 1$.

**Theorem 5** *The greedy strategy is an $O(\sqrt{\log n})$-approximation to the FTP in the plane. One can compute the greedy awakening tree in time $O(n \log^{O(1)} n)$.*

**Theorem 6** *The Greedy strategy is an $O((\log n)^{1-1/d})$ approximation to the FTP in $d$ dimensions. One can compute the greedy awakening tree in time $O(n \log^{O(1)} n)$.*

In both theorems, the algorithmic complexity follows from applying dynamic methods for nearest neighbor search (or approximate nearest neighbor search [3, 5]) in order to identify which asleep robot is closest to a newly awakened robot. The nearest neighbor search requires deletions, since robots get deleted from the set of candidate targets when they awaken.

We concentrate on proving the approximation bound for the 2-dimensional case (Theorem 5); the $d$-dimensional case is a fairly direct generalization.

Suppose that we are given a greedy awakening tree for a swarm of size $|S| = n$. We can convert this tree into an awakening tree (at no increase in makespan) having the following property: on any root-to-leaf path there is at most one non-leaf node having out-degree 1 (rather than 2). Based on this property, and a simple counting argument, there exists a root-to-leaf path, $P = (p_0, p_1, \ldots, p_K)$, having $K \leq \log n$ edges of lengths $r_i = \text{dist}(p_i, p_{i+1})$. At the time $t$ when the last node $p_K$ is reached, all of the remaining asleep robots are robots that will be awakened by other branches of the awakening tree (by definition of the awakening tree). The makespan, therefore, must be at most $t + D \leq t + 2R$; otherwise, the robot at $p_K$, the last robot on the branch, would awaken one of the remaining asleep robots (i.e., the last robot would not have been last on its branch). We will show that $t = O(\sqrt{\log n} \cdot t^*)$, implying that the makespan is $O(\sqrt{\log n} \cdot t^*)$

Fixing attention now on one root-to-leaf path, $P$, in the awakening tree, we define the *outer circle* $C_i$ to be the circle centered at $p_i$ with radius $r_i = \text{dist}(p_i, p_{i+1})$; the *inner circle* $c_i$ is the circle centered at $p_i$ with radius $\frac{1}{3} r_i$. The properties of the greedy heuristic ensure the following claim:

**Claim 7** *None of the points $p_{i+1}, \ldots, p_K$ lie inside circle $C_i$.*

**Proof:** If some point $p_j \in C_i$, for some $j > i + 1$, then the greedy strategy would dictate going next to the closest such point to $p_i$, rather than going next to $p_{i+1}$.

□

The proof of Theorem 5 is based on an *area-covering argument*. Specifically, we provide a bound on the area covered by the circles $C_0, C_1, \ldots C_{K-1}$, showing that

$$\sum_{i=0}^{K-1} r_i^2 = O(R^2),$$

where $R$ is the radius of the swarm. The subtlety of the proof is that neither the outer circles $C_0, \ldots, C_{K-1}$ nor even the inner circles $c_0, \ldots, c_{K-1}$ are disjoint. The proof is based on the following lemma:

**Lemma 8** *The combined area covered by the (inner or outer) circles is $O(R^2)$; that is,*

$$\sum_{i=0}^{K-1} area(C_i) = O\left(\sum_{i=0}^{K-1} area(c_i)\right) = O\left(\sum_{i=0}^{K-1} r_i^2\right) = O(R^2).$$

Before proving Lemma 8, we show how to use this lemma to prove Theorem 5.

**Proof of Theorem 5:** The length, $L$, of the root-to-leaf path $P$ is simply $L = \sum_{i=0}^{K-1} r_i$. By the Cauchy-Schwartz inequality[1], the fact that $K \leq \log n$, and Lemma 8 we get

$$L = \sum_{i=0}^{K-1} r_i \leq \sqrt{K}\sqrt{\sum_{i=0}^{K-1} r_i^2} = O(R\sqrt{\log n}).$$

Since this result holds for any root-to-leaf path in the awakening tree, the approximation bound follows. □

**Proof of Lemma 8:** Each (outer,inner) circle pair, $(C_i, c_i)$, is assigned to a *circle class* according to its radius. Without loss of generality, let the smallest outer circle have radius 1. Define *class i* to be the set of (outer,inner) circle pairs such that the radius of the outer circle has radius $r$, with $2^{i-1} \leq r < 2^i$.

While pairs of outer circles (and pairs of inner circles) may overlap, using the property of circle classes, we show the following claim:

**Claim 9** *Any pair of inner circles belonging to the same class are disjoint.*

**Proof:** Let $p_j$ and $p_k$ be two points on path $P$. Suppose that the circles associated with $p_j$ and $p_k$ belong to the same circle class. Without loss of generality, assume that $p_j$ occurs before $p_k$ on $P$. By Claim 7, since $p_k$ is reached after $p_j$ along the branch $P$ of the awakening tree, $\text{dist}(p_j, p_k) > r_j$. Since $C_j$ and $c_j$ belong to the same circle class, we know that $r_k < 2r_j$, implying that the radius of $c_k$, $\frac{1}{3}r_k$, is less than $\frac{2}{3}r_j$. Therefore, since $\text{dist}(p_j, p_k) \geq r_j$, the inner circles $c_j$ and $c_k$ are disjoint. □

Let area $A_i$ be the area covered by inner circles of class $i$. We claim that

**Claim 10** *In order for the inner circles of class i associated with path P to cover area $A_i$, the corresponding edges of P (associated with circle pairs of class i) must have total length $\ell_i$ satisfying*

$$\frac{9A_i}{2^{i+1}\pi} \leq \ell_i \leq \frac{9A_i}{2^{i-2}\pi}.$$

**Proof:** In order to cover area $A_i$ we need at least $\frac{A_i}{(\frac{2^i}{3})^2\pi}$ circles, since each inner circle of class $i$ has radius at most $\frac{2^i}{3}$. The length of each edge corresponding to a class $i$ circle pair is at least $2^{i-1}$; thus, the total length of these edges is at least $\frac{9A_i}{2^{i+1}\pi}$. Similarly, we have at most $\frac{A_i}{(\frac{2^{i-1}}{3})^2\pi}$ circles, since the (non-overlapping) inner circles of class $i$ each have radius at least $\frac{2^{i-1}}{3}$. The length of each edge corresponding to a class $i$ circle pair is at most $2^i$; thus, the total length of these edges is at most $\frac{9A_i}{2^{i-2}\pi}$.

[1] Here, we apply the Cauchy-Schwartz inequality, $|\mathbf{r} \cdot \mathbf{s}| \leq \|\mathbf{r}\| \cdot \|\mathbf{s}\|$, with vectors $\mathbf{r} = (r_0, r_1, \ldots, r_{K-1})$ and $\mathbf{s} = (1, 1, \ldots, 1)$.

□ Since along path $P$ of the awakening tree we have circles of different classes appearing, not necessarily in order of size, we may have inner circles overlapping, thus not covering "new" area. We need the following claim:

**Claim 11** *The length of $P$ that does* not *correspond to edges whose inner circles cover new area is at most a constant fraction of the length that corresponds to edges whose inner circles do cover new area.*

**Proof:** Consider the worst case, in which all area covered by class-$i$ inner circles is covered in all larger classes $j > i$. In class $i$ the average amount that we must walk along $P$ in order to cover 1 unit of area is between $\frac{9}{2^{i+1}\pi}$ and $\frac{9}{2^{i-2}\pi}$. Therefore suppose that a unit of area is covered first in class $i$, where average cost is between $\frac{9}{2^{i+1}\pi}$ and $\frac{9}{2^{i-2}\pi}$. All of the cost for covering this area in large classes sums to at most

$$\sum_{j>i} \left(\frac{1}{2}\right)^{j+1} \frac{9}{\pi} \leq \left(\frac{1}{2}\right)^{i+1} \frac{9}{\pi}.$$

□

Thus, by Claim 11, the distance traveled along $P$ in which no new area is covered is of the order of the distance traveled in which new area is covered. Since the total area covered is $O(R^2)$, the claim of Lemma 8 follows. □

## 2.2 Other Heuristic Wakeup Strategies

The greedy strategy has the following weakness: it may be preferable for a robot to travel a longer distance to obtain a better payoff. (See Figure 3.) In this section we examine alternative strategies in an attempt to overcome this weakness.

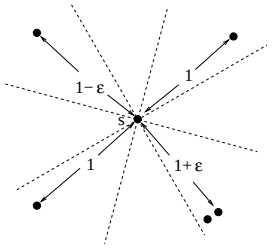


Figure 3: Example in which greedy is suboptimal: Greedy initially sends the robot at $s$ to the northwest corner, while a better strategy (maximizing the "bang-for-the-buck") sends the robot at $s$ initially to the southeast corner, where two nearby robots are awakened at essentially the same time, leading to a better overall wakeup strategy (makespan 3 versus $1 + 2\sqrt{2}$).

We design our alternative strategies while keeping in mind the actual application that motivated our study: the need to activate a swarm of small experimental robots, each equipped with certain sensors. The sensors on the actual robots in our project (joint with HRL Labs) have the feature that they sense other robots (or obstacles) in each of $k$ sectors, evenly distributed around the (circular) robot. (Our robots have $k = 8$, implying 45-degree sectors.) Within each of its sectors, a robot can detect only the closest other robot. (In fact, it can sense another robot only within a limited range; we do not model this constraint here.) Thus, there are at most $k$ options facing a robot once it is activated: Which sector should be selected? Once the sector is selected, the robot

heads for the closest asleep robot it has sensed in that sector. A greedy strategy that uses sensor sectors will select the sector whose closest robot is closest.

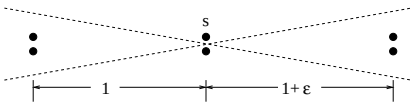


Figure 4: Example in which both greedy and bang-for-the-buck strategies are suboptimal: Both strategies use the awake robot at $s$ to awaken the nearby robot and then send both robots to the left, while an optimal strategy would send one robot to the left, one to the right.

**Bang-for-the-Buck.** A natural strategy, which we call "bang-for-the-buck", is to choose the next target to awaken based on maximizing the ratio of "value" ("bang") to "cost" ("buck"). There are a variety of heuristic measures of potential value; a particularly simple one is to consider the value of a sector to be the number of currently asleep robots in the sector. (Note that this is a quantity that our actual robots may not be able to detect, except approximately, since they can only reliably ascertain the presence and approximate position of the closest robot in a sector.) The cost of a sector is naturally chosen to be the distance to the nearest asleep robot in the sector. In Figure 3, the bang-for-the-buck strategy is significantly better than greedy; Figure 4 shows an example in which both greedy and bang-for-the-buck may be suboptimal.

**Random Sector Selection.** The goal of this strategy is to "mix up" at random the directions in which robots head to awaken other robots. When a robot awakens, it selects, at random, a sector and chooses its next target to be the closest asleep robot in that sector.

**Opposite Cone.** In this strategy the goal is to enforce a certain amount of "mixing up" of directions that robots head to awaken new targets, by sending a newly awakened robot in a nearly opposite direction from that of the robot that awakened it. In particular, suppose robot A heads due east to awaken robot B at point $p$; then, the next target that robot A selects will be chosen to be the closest asleep robot in a cone centered on a vector to the west, while the target for robot B to awaken next will be selected from a cone centered on a vector to the east.

# 3 Experiments

## 3.1 Experimental Setup

Our experiments are based on a Java simulation of our various strategies. All tests were performed on a PC running Linux OS. The graphical user interface permits the user to select the choice of strategy, the parameters, and the input dataset, and it optionally shows a graphical animation of the simulation.

**Datasets.** We tested our strategies on both geometric and non-geometric datasets. We investigated four classes of geometric datasets:

1. Uniform: The $n$ points that represent the originally asleep robots are generated uniformly at random over a large square (600-by-600) that represents the environment.

2. Cluster: We generate $\sqrt{n}$ clusters, each of random size having mean $\sqrt{n}$. Each cluster is generated uniformly over a square of side length $c$, whose upper left corner is uniformly distributed over the large (600-by-600) square that represents the environment. In our experiments we chose $c = 2\sqrt{n}$.

3. Grid: The $n$ asleep robots are placed at the points of a $p$-by-$q$ regular (rectangular) grid. For our experiments, we used the same spacing in $x$ as in $y$.

4. Hexagonal grid: The $n$ points are placed according to a regular hexagonal grid.

5. TSPLIB: The points come from symmetric traveling salesman instances in the TSPLIB [7] that have data of type EUC_2D (68 instances).

Since our non-greedy strategies are specified geometrically, they were applied only to the geometric data.[2]

The greedy strategies were applied to *non*-geometric datasets, including:

1. Star metrics: $n-1$ asleep robots are positioned at the leaves of a star, with the source robot at the root of the star. Each leaf is at the end of a spoke of random length (we chose to be uniform between 1 and $n$, unless otherwise noted). Distances are measured according to path length in the star. We consider stars having exactly one asleep robot per leaf (case "1-1") and stars having potentially many asleep robots per leaf (case "1-m"). In case 1-m, we generate $O(\sqrt{n})$ spokes and randomly assign a number $m$ of robots to each spoke, with $m$ chosen uniformly between 1 and $O(\sqrt{n})$.

2. TSPLIB: In order to consider also instances that come from more general (non-geometric) metrics, we selected instances from the TSPLIB, using the symmetric traveling salesman files, with data of type MATRIX (14 instances).

Adapting our algorithms to non-geometric data, we needed to address the problem of *stopping* a robot that is in motion, along an edge of the network, towards a chosen target. (This is not an issue if we are using the delayed target choice option, since then a robot only moves once it is able to move all the way to its chosen target.) Since these datasets constitute an abstract metric space (network with edge lengths), robots do not have geometric coordinates associated with their current locations. Thus, when considering a reassessment of the target for that robot (during a refresh event), we must be able to compute distances from the robot's current position, somewhere along an edge, to each of the possible choices of target. This is done by considering the robot's position to be a point along the edge, interpolated according to the fraction of the edge length already traversed; then, the distance from the robot's current position to a potential target is computed accordingly.

**Performance Measures.** We maintain performance statistics, including: (1) *total time* of the simulation; (2) *total distance* traveled by all robots; and (3) *average distance* traveled by robots that do any traveling. The radius, $R$, of the swarm is the distance from the source robot to the farthest away initially asleep robot; $R$ is, of course, a lower bound on the total time of the simulation. (Recall that we assume robots travel at speed 1.) We found that total distance and average distance were tightly correlated with the total time of the simulation; thus, here we report results only on the total time.

[2] While our non-greedy strategies can be generalized to non-geometric datasets, this has not been a part of our experiments so far.

**Parameter Choices.** For each of the strategies, we considered each possible setting of the set of parameter choices (Claims, Refresh, or Delayed Target Choice) discussed in Section 2.1:

**Claims:** If this parameter is set to *true*, once an awake robot selects its next target, it generates a claim on that sleeping robot, and no other robot can select it. (The tie-breaking procedure depends on which wakeup strategy is used.) If the parameter is set to *false*, a robot selects its target only according to the wakeup strategy, disregarding possible ties or multiple robots going towards the same target.

**Refresh:** If the refresh parameter is set to *true*, then, at each *event* (time instant when a robot is awakened), the matching between awake robots and asleep robots is recomputed (using the greedy matching strategy described earlier). If the parameter is set to *false*, then once a robot $A$ selects a target $B$, $A$ will not change this target selection until $A$ reaches $B$ or until some other robot $C$ reaches $B$ (at which point $A$ selects a new target).

**Delayed Target Choice:** This further refinement allows robot to stay on the same spot when their allowed step distance is not enough to reach a sleeping robot. The allowed step distance is determined by the shortest distance between all the pairs of awaken robots and its selected targets. The robot will accumulate units of distance that will be used towards its target when the amount of units accumulated plus the step distance allows it to reach that target.

### 3.2 Experimental Results

The experiments on synthetically generated datasets were conducted as follows. For each choice of wakeup strategy, parameters, and dataset, a set of 100 runs was performed, with 10 runs for each value of $n \in \{100, 200, 300, \ldots, 1000\}$, where $n$ is the number of asleep robots at the generated points of the dataset.

The experiments on datasets from the TSPLIB (EUC_2D or MATRIX) were done once per dataset; one asleep robot was placed initially at each point of the dataset.

We performed runs for the following combinations of strategies and datasets: Greedy was run on all datasets (Uniform, Cluster, Grid, Hexagonal Grid, Stars 1-1, Stars 1-m, TSPLIB (EUC_2D and MATRIX)), while Bang-for-the-Buck, Random Sector Selection, and Opposite Cone were run on only the geometric instances (Uniform, Cluster, Grid, Hexagonal Grid, TSPLIB (EUC_2D)).

In our plots, the horizontal axis corresponds to the swarm size $n$, the vertical axis to the ratio of the makespan to the lower bound on makespan (the radius, $R$, except in some star-metric cases). Thus, the vertical axis shows an upper bound on the approximation ratio.

**Parameter Choices.** We experimented with various parameter choices over a common dataset. The claims option is very significant; without it the robots travel in groups rather than spreading out. There is a significant advantage in using the refresh option. While less significant, the delayed target choice is also advantageous. Figure 5 shows the result of running greedy on uniformly distributed points; other datasets yield similar results, with the delayed target choice showing a more pronounced advantage in the case of cluster datasets.

**Wakeup Strategy Comparison.** The main conclusion we draw from our experiments is that the greedy strategy is a very good heuristic, most often outperforming the other strategies in our comparisons. See Figures 6, 8, and 9. As the size ($n$) of the swarm increases, the approximation ratios tend to stay about the same or decrease; we suspect this is because $R$ becomes a better lower bound for larger swarms. The upper bounds (Time/$R$) on approximation factors in the geometric instances are between 1.0 and 1.5. For star metrics, the ratio Time/$R$ is significantly

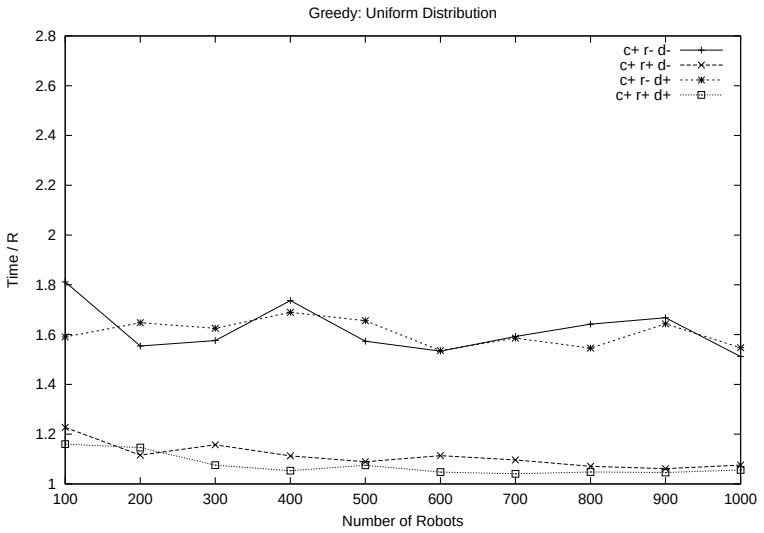


Figure 5: Greedy on uniform distributions: Choices of parameters for claims (c), refresh (r), and delayed target choice (d). A "+" ("-") indicates the parameter is set to *true* (*false*). Using refresh is seen clearly to be desirable; among those using refresh, using the delayed target choice is seen to be somewhat better.

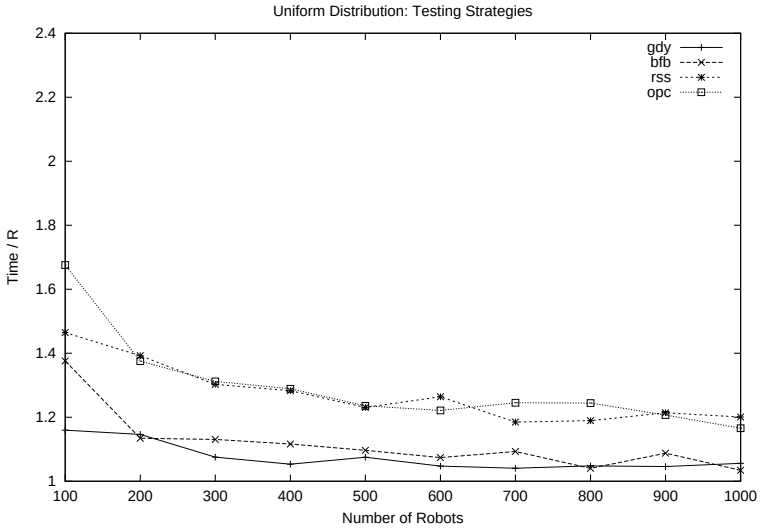


Figure 6: Greedy (gdy), Bang-for-the-Buck (bfb), Random Sector Selection (rss), and Opposite Cone (opc) tested on uniformly distributed swarms. The four strategies give a constant approximation to the lower bound with values in between 1 and 1.5. Greedy gives better approximation.

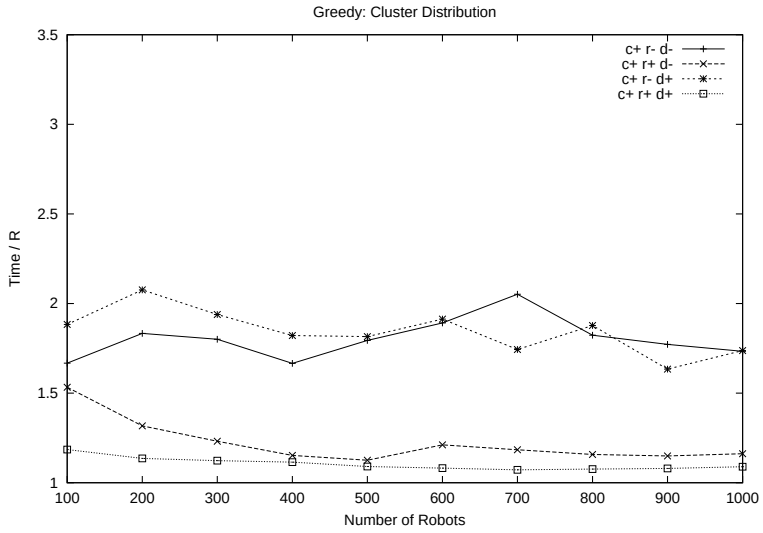


Figure 7: Greedy on cluster distributions: Different combinations of choices of parameters for claims (c), refresh (r), and delayed target choice (d). A "+" ("-") indicates the parameter is set to *true* (*false*). As in the case of uniform distributions, using refresh is seen clearly to be desirable; among those using refresh, using the delayed target choice is seen to be somewhat better (and the improvement is more than was seen in the uniform case).

higher (between 2 and 3), but this is due to the fact that $R$ is a poor lower bound in the case of stars. In order to verify this, we computed an alternative lower bound specifically for star metrics having one robot per leaf. (The lower bound is $2L_{min}(\lceil \log(n+1) \rceil - 1) + L_{max}$, where $L_{min}$ (resp., $L_{max}$) is the length of the shortest (resp., longest) spoke of the star. More complex lower bounds can be similarly derived for the case of stars with many robots per leaf.) Figure 11 shows the results of greedy on stars 1-1 datasets (of various spoke length distributions) using this lower bound in computing the approximation ratio; we see that there is a striking improvement over using the lower bound of $R$ (which is particularly poor for star metrics). We also give tables (Tables 1,2) showing the percentage of wins for each strategy, and, finally, report in Table 3 the results for greedy on the non-geometric TSPLIB instances.

| **Algorithm** | **Wins** | **%** |
|---|---|---|
| GDY | 45 | 66.20 |
| BFB | 22 | 32.35 |
| RSS | 1 | 1.45 |
| OPC | 0 | 0.00 |

| **Algorithm** | **Approx. Rate** | | | **Winning %** | | |
|---|---|---|---|---|---|---|
| | **Best** | **Worst** | **Avg** | **Max** | **Min** | **Avg** |
| GDY | 1.00 | 1.29 | 1.06 | 41.13 | 0.01 | 6.20 |
| BFB | 1.00 | 1.36 | 1.07 | 18.71 | 2.24 | 3.97 |
| RSS | 1.04 | 1.04 | 1.04 | 2.24 | 2.24 | 2.24 |

Table 1: Left: Comparing strategies on the 68 TSPLIB (EUC_2D) datasets: Greedy (GDY), Bang-for-the-Buck (BFB), Random Sector Selection (RSS), and Opposite Cone (OPC). Greedy outperforms the other strategies 2 out of 3 runs. Right: Winning strategies for the TSPLIB (EUC_2D) datasets: For those runs in which a strategy outperformed the others, we compute the maximum, minimum and average approximation factor and percent by which it lead over the second-place strategy.

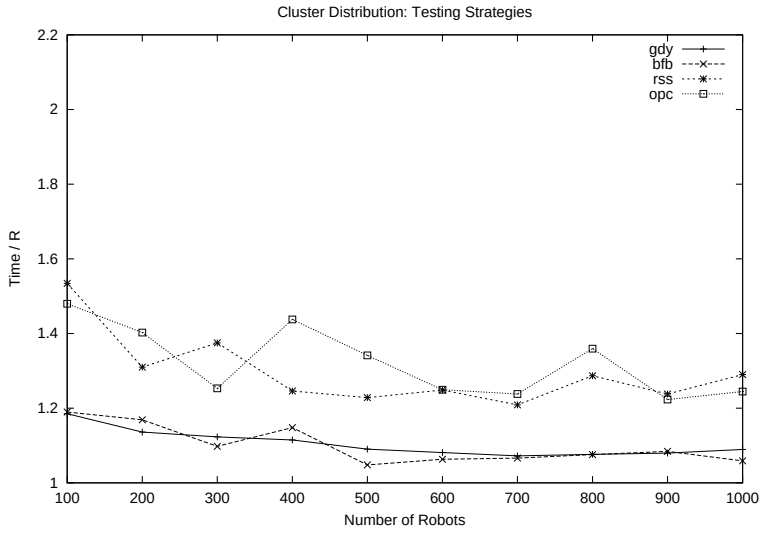


Figure 8: Strategy comparison on cluster data. Greedy generally outperforms the other strategies, with bang-for-the-buck holding a close second. The approximation factors are essentially constant, with similar values to the uniform datasets.

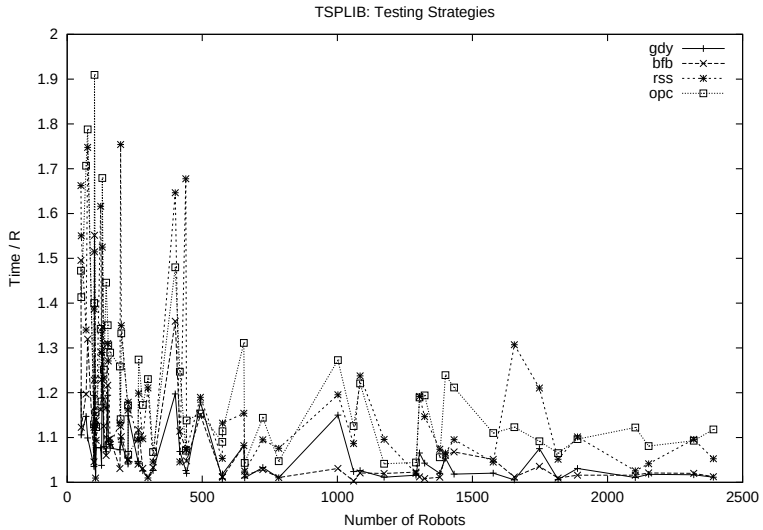


Figure 9: Strategy comparison on TSPLIB EUC_2D data.

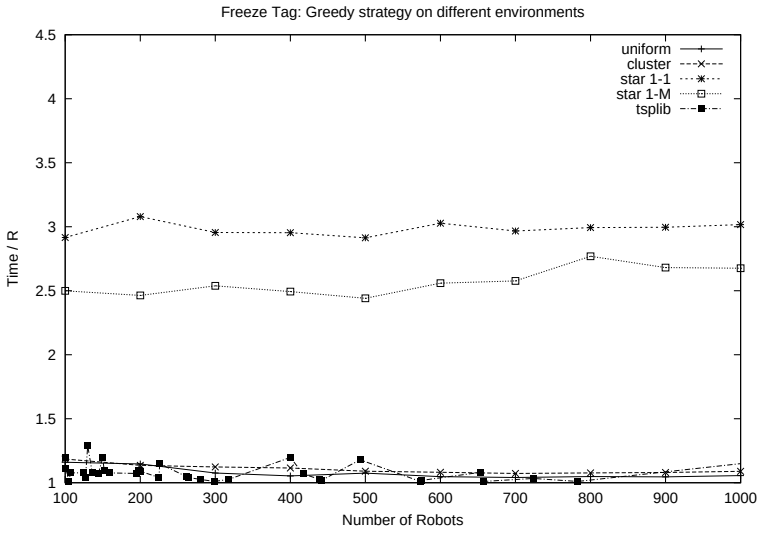


Figure 10: Comparing greedy on five different datasets. Note that the approximation factor stays relatively flat (constant) with swarm size $n$. While the factor is between 1.0 and 1.3 for the uniform and cluster cases and TSPLIB (EUC_2D), it grows to 2.5-2.7 for the case of stars with multiple robots at each leaf, and to about 3 for the case of stars 1-1 (1 robot per leaf). It is interesting to note that the TSPLIB results are indistinguishable from the uniform and cluster data results.

# 4 Conclusion

We have done theoretical and experimental analysis of variants of the greedy strategy, and other heuristics, for the freeze-tag problem. Our theoretical analysis is tight, showing that greedy per-

| **Algorithm** | **Win %** |
|---|---|
| GDY | 62 |
| BFB | 37 |
| RSS | 0 |
| OPC | 1 |

| **Algorithm** | **Approx. Rate** | | | **Winning %** | | |
|---|---|---|---|---|---|---|
| | **Best** | **Worst** | **Avg** | **Max** | **Min** | **Avg** |
| GDY | 1.01 | 1.31 | 1.06 | 57.32 | 0.33 | 8.83 |
| BFB | 1.00 | 1.16 | 1.04 | 27.23 | 0.13 | 4.04 |
| OPC | 1.23 | 1.23 | 1.23 | 5.25 | 5.25 | 5.25 |

Table 2: Comparing strategies on uniform datasets. Left: Comparing strategies for uniform datasets: Greedy (GDY), Bang-for-the-Buck (BFB), Random Sector Selection (RSS), and Opposite Cone (OPC) tested over 100 randomly generated swarms of robots, uniformly distributed in a square. Greedy outperforms the other strategies in almost 2 out of 3 runs. Right: Winning strategies for uniformly distributed robots: For those runs in which a strategy outperformed the others, we compute the maximum, minimum and average approximation factor and percent by which it lead over the second-place strategy.

| **# of Robots** | 17 | 21 | 24 | 26 | 42 | 42 | 48 | 48 | 58 | 120 | 175 | 535 | 561 | 1032 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Approx. Rate** | 1.06 | 1.04 | 1.16 | 1.36 | 1.08 | 1.12 | 1.03 | 1.24 | 1.19 | 1.12 | 3.26 | 3.01 | 1.17 | 3.19 |

Table 3: Results of greedy strategy for TSPLIB MATRIX (non-geometric) instances. When the number of robots is small, the approximation ratio is similar to the case of geometric environments, but as the number of robots starts to increase, some ratios get closer to the values attained for star metrics (using the relatively poor lower bound of $R$).

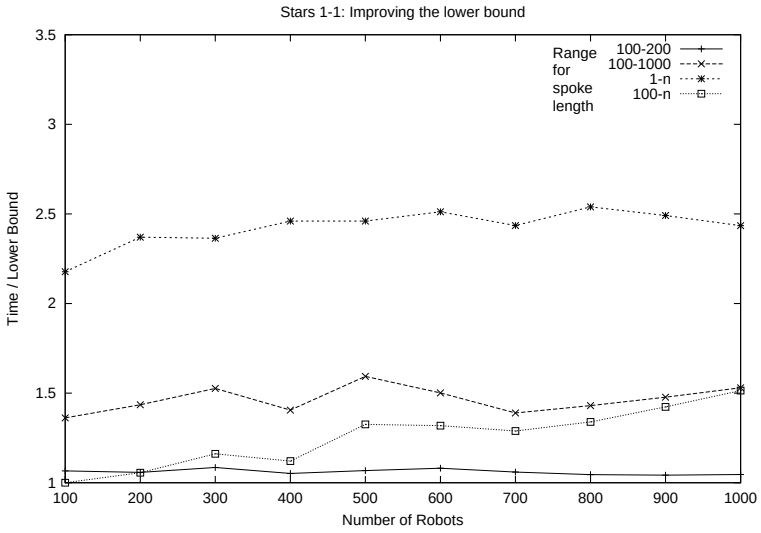


Figure 11: Comparing greedy on star 1-1 datasets, *using the improved lower bound* of $2L_{min}(\lceil \log(n+1) \rceil - 1) + L_{max}$. Each curve corresponds to a randomly generated dataset having spoke lengths uniformly generated in the indicated interval. Note that for spoke lengths that are generated in the length interval (100,200), the approximation factor is essentially 1.

forms within a factor $\Theta((\log n)^{1-1/d})$ of optimal. Our experiments show, however, that greedy (and other heuristics) perform very well on a broad range of datasets, yielding a small constant-factor approximation. In continuing and future work, we plan to analyze hybrid strategies that combine the best features of greedy and bang-for-the-buck. We also are extending our improved lower bounds for stars with one robot per leaf (1-1 case) to the 1-m case. From a theoretical point of view, our main goal is to obtain an $o(\log n)$-approximation for general metric spaces.

**Acknowledgements.** We thank Doug Gage for discussions motivating this research. We thank Gregory Bachelis for useful discussions on the upper bound analysis of greedy. This research was partially supported by grants from HRL Labs (DARPA subcontract), NASA Ames Research, the National Science Foundation, Sandia National Labs, and the U.S.-Israel Binational Science Foundation.

# References

[1] 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*, pp. 568–577, 2002.

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

[3] S. N. Bespamyatnikh. Dynamic algorithms for approximate neighbor searching. In *Proc. 8th Canad. Conf. Comput. Geom.*, pp. 252–257, 1996.

[4] S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman. A Survey of Gossiping and Broadcasting in Communication Networks. *NETWORKS*, 18:319–349, 1988.

[5] S. Kapoor and M. Smid. New techniques for exact and approximate dynamic closest-point problems. *SIAM J. Comput.*, 25:775–796, 1996.

[6] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. *Proc. 35th Ann. Sympos. on Foundations of Computer Sci.*, pp. 202–213, 1994.

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