Theoretical and Experimental Analysis of Heuristics for the "Freeze-Tag" Robot Awakening Problem

Marcelo O. Sztainberg, Esther M. Arkin, Michael A. Bender, and Joseph S. B. Mitchell

***Abstract*—In the "freeze-tag" problem, we are given a swarm of n sleeping (frozen or inactive) robots and a single awake (active) robot. The goal is 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 acceptably 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 a tight performance bound of $\Theta((\log n)^{1-1/d})$ in d dimensions. The geometric case contrasts 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 factor of $o(\log n)$.**

***Index Terms*—Approximation algorithms, computational geometry, emergent behavior, multiple robots, optimization, swarm robotics.**

I. Introduction

WE CONSIDER a 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 the sleeping robot and touches it; then, the newly awake robot joins the set of active robots in awakening other sleeping robots. Our goal is to activate (wake up) all of the robots as quickly as possible, i.e., we want to minimize the makespan, the time when the last robot is awakened.

This problem has been coined the "freeze-tag" problem (FTP) [4] because of its similarity to 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. 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 [11] for a survey and [6] and [13] 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 an out-degree of at most two.

What makes the FTP particularly intriguing is that *any* reasonable ("nonlazy") strategy yields an $O(\log n)$-approximation [4, Prop. 1.1], whereas no strategy is known for general metric spaces that yields a $o(\log n)$-approximation.[1] (Some recent improvements for special cases are reported in [5].) We say that a strategy is nonlazy if each awake robot claims and goes to a sleeping unclaimed robot, if one exists, at the moment that he awakes; if no sleeping unclaimed robot exists, then a newly awakened robot does not move. Arkin *et al.* [4] show that even 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 an $O(1)$-approximation is possible, and an $o(\log n)$-approximation for the special case of *ultrametrics*, in which the underlying metric is defined by a rooted tree, all robots are initially at the leaves, the length of each root to leaf path is the same, and robots must travel along edges of the tree.

Our Results: We provide both theoretical and experimental results.

1) We prove 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

Manuscript received July 23, 2002; revised March 3, 2003. This paper was recommended by Associate Editor N. Amato and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported in part by the Honda Fundamental Research Laboratory, in part by HRL Labs through a DARPA subcontract, in part by Metron Aviation, in part by NASA Ames Research under Grant NAG2-1325, in part by the National Science Foundation under Grant CCR-0098172, Grant EIA-0112849, and Grant CCR-0208670, in part by Sandia National Labs, and in part by the U.S.–Israel Binational Science Foundation under Grant 2000160. This paper was presented in part at the Eighth Scandinavian Workshop on Algorithm Theory (SWAT 2002), Turku, Finland, July 3–5, 2002.

M. O. Sztainberg is with the Computer Science Department, Northeastern Illinois University, Chicago, IL 60625 USA (e-mail: M-Sztainberg@neiu.edu).

E. M. Arkin and J. S. B. Mitchell are with the Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600 USA (e-mail: estie@ams.sunysb.edu; jsbm@ams.sunysb.edu).

M. A. Bender is with the Department of Computer Science, State University of New York, Stony Brook, NY 11794-4400 USA (e-mail: bender@cs.sunysb.edu).

Digital Object Identifier 10.1109/TRO.2004.829439

[1]Please see note on new development, after the Conclusion.

the plane, the greedy heuristic yields an $O(\sqrt{\log n})$-approximation. We prove that this analysis is tight by exhibiting matching lower bounds.

While better approximations are known for the geometric instances (indeed, there is an efficient PTAS for any constant d dimensions [4]), the greedy strategy is algorithmically simple and "myopic," in that each robot decides which robot to awaken next without needing to plan ahead.

2) We perform an experimental investigation of heuristic strategies for the FTP, comparing different design choices for greedy strategies and comparing them with other heuristics. We present experimental results on classes of randomly generated data, as well as on datasets from the TSPLIB, a repository of instances for the Traveling Salesperson Problem.

Other Related Work: In addition to the original work on the FTP and the minimum broadcast-time problem already mentioned, there is a vast array of literature on algorithms for robot swarms. Among the most studied problems in this field are environment exploration [2], [3], [8], [10], [12], [21], [23], robot formation [14]–[17], searching [22], and recruitment [20], [23]. There has also been considerable research in the area of algorithms for multiagent problems directly inspired by ant behaviors, such as searching and covering [20]–[23].

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 sleeping robots are located. We let $s(\notin S)$ 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., $R = \max_{p \in S} \operatorname{dist}(s,p)$. We let $D = \max_{p,q \in S \cup \{s\}} \operatorname{dist}(p,q)$ 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.

II. Wakeup Strategies for the FTP

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

A. Greedy Strategies

A natural strategy for the FTP is the greedy strategy, in which an awake robot chooses the nearest sleeping 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, and so on.

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: When a robot is awakened, it lays a *claim* on the sleeping robot that it intends to awaken next. We distinguish between *claims with full communication* and *claims with local communication*. In the full-communication model, when a robot awakens and claims a sleeping robot, no other robot is allowed to claim it (ties are negotiated and broken arbitrarily); this model assumes that the robots have global communication or some means of marking a robot as "claimed." (This is the version implemented in [18].) In the claims with the local-communication model (implemented and reported here), when a robot is awakened by another robot, only these two robots negotiate in order to avoid laying claim to the same robot; the sleeping robots that they "claim" are, in fact, available to others to claim and may have already been claimed before (but due to their local information, they have no way of knowing this).

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 sleeping 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 a sleeping 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 sleeping 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 sleeping 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 oscillation happens when an awake robot chooses a sleeping target, but before the robot reaches its target, another robot claims the target. With the *delayed target choice* option, we avoid these oscillations by using some amount of look-ahead. Specifically, we avoid committing to the direction a robot is heading until that robot has traveled far enough to awaken a target, at which point the target (and its position) is fully determined.

Thus, the greedy strategy proceeds as follows. For each robot i, we store the last position p_i of the awakening event in which i participated (either as "awakener" or "awakenee"), together with its accumulated distance (equivalently, the time) δ_i of travel since that event. Then we compute matchings greedily between the p_i's for awake robots and the sleeping robots, using $\operatorname{dist}(p_i, p_j) - \delta_i$ as the edge weight between p_i and the location p_j of sleeping robot j. After each awakening event, we

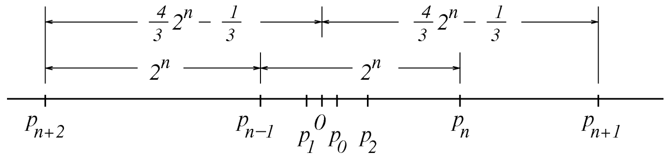

Fig. 1. Lower bound construction in $\Re^1$, for which greedy is at least $4 - \epsilon$ times optimal.

recompute the matching. Thus, only the minimum-weight edge of the matching needs to be computed at each event.

Remark: Each of the above options requires certain assumptions about the robot model. Claims with full communication assumes that, if robot A wants to claim robot B (implying that robot A can sense the presence of B), then any other robot C that wants to claim robot B is able to communicate with robot A. Refresh requires greater communication among robots, since robots must negotiate more frequently. Delayed target choice requires look-ahead: the robot swarm (or a central command, which is in communication with the swarm) must plan before movement begins.

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 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 sleeping robots at points $S = \{p_0, p_1, \dots, p_n, p_{n+1}, p_{n+2}\}$, 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 $j = 1, 2, \dots, n$. Note that $p_{2k} = (2/3)2^{2k} + (1/3)$ and $p_{2k+1} = -(2/3)2^{2k+1} + (1/3)$, so $p_{n-1} = -(1/3)2^n + (1/3)$ and $p_n = (2/3)2^n + (1/3)$. Finally, we place 2^{n+2} sleeping robots at point $p_{n+2} = p_{n-1} - 2^n = -(4/3)2^n + (1/3)$ and we place 2^{n+1} sleeping robots at $p_{n+1} = -p_{n+2} = (4/3)2^n - (1/3)$. Refer to Fig. 1.

The greedy strategy sends the source robot to the right, a distance of one, to awaken the one robot at p_0, then two robots to the left, a distance of two, to awaken the two robots at p_1, then four robots to the right, a distance of four, to awaken the four robots at p_2, and so on. (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 $p_{2k} = (2/3)2^{2k} + (1/3) - \delta^{2k+1}$ and $p_{2k+1} = -(2/3)2^{2k+1} + (1/3) + \delta^{2\ell+2}$.) 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 $(2/3)2^n - (2/3)$, to awaken the robots at p_{n+1}, and then finally, all of the robots go to the *left* a distance of $(8/3)2^n - (2/3)$ to p_{n+2}. The makespan is given by the total distance traveled by the source robot, which is $(2 \cdot 2^n - 1) + ((2/3)2^n - (2/3)) + ((8/3)2^n - (2/3)) = (16/3)2^n - (7/3)$.

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 $p_0, p_2, p_4, \dots, p_{n-2}, p_n, p_{n+1}$, while the robot that was awakened at p_0 heads to the *left* to awaken all other robots $(p_1, p_3, \dots, p_{n-1}, p_{n+2})$. The makespan is $1 + (4/3)2^n - (1/3) = (4/3)2^n + (2/3)$.

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

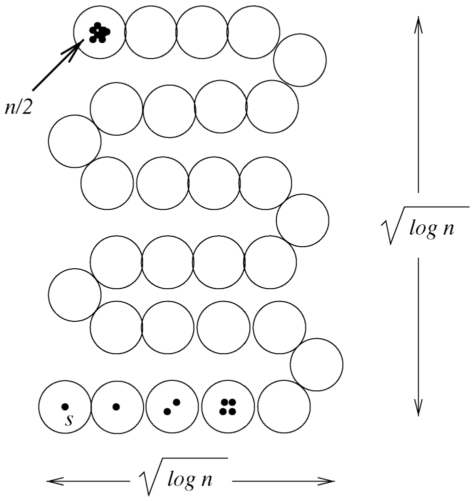

Fig. 2. Lower bound construction in $\Re^2$.

Theorem 2: The greedy heuristic is an $\Omega(\sqrt{\log n})$-approximation to the FTP in the plane. The greedy strategy is an $\Omega((\log n)^{1-1/d})$-approximation in $\Re^d$.

Proof: We begin with the proof in the plane ($\Re^2$). We arrange $\log n$ disks, each with radius one, along a zigzag path having $\sqrt{\log n}$ rows, each having $\sqrt{\log n}$ disks, with disks touching but not overlapping along the path. Refer to Fig. 2. The source robot is at the center s of the first disk. There is one sleeping robot at the center of the second disk, then two at the center of the next, then four, and so on, with the number of sleeping robots at the center of each disk doubling as we advance along the path of touching disks. The last disk has (about) $n/2$ robots. (More precisely, we consider values of n of the form $n = 2^{m^2-1} - 1$, for integer m. Then, the zigzag 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 sequence along the zigzag 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 two. A better strategy, however, sends the source robot vertically upwards,

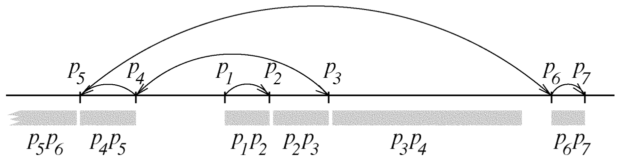

Fig. 3. Root-to-leaf path $(p_1, p_2, p_3, \ldots)$ in the greedy awakening tree, with the covering intervals shaded for each step. Steps p_1p_2, p_2p_3, p_4p_5, and p_6p_7 fall in case 1; steps p_3p_4 and p_5p_6 fall in case 2.

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 in the plane.

The proof generalizes to $\Re^d$ as follows. 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 sleeping robots at the centers of the disks are 1, 2, 4, 8, ..., and the one source robot is at the center of the first disk. As in the two-dimensional (2-D) case, the greedy strategy awakens robots in order along the path, taking a total of $\log n$ steps, each of size about two. 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) 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 1-D case.

Theorem 3: The greedy heuristic for a swarm $S \subset \Re$ of points on a line $(d = 1)$ is a 4-approximation.

Proof: Consider the longest path $p_1, p_2, \ldots$ in the awakening tree. With each step from p_i to p_{i+1} of length r_i along the path, we associate or *cover* an open interval of equal length r_i that is "charged" with the step. Roughly speaking, whenever a region is covered, we know that it has been cleared of sleeping robots. More precisely, whenever a region is covered, it has been cleared of sleeping robots that are awakened on this path $p_1, p_2, \ldots$ in the awakening tree, but sleeping robots could still remain in the covered region if they are awakened along other paths down the awakening tree.

Assume that the step from p_i to p_{i+1} is to the *right*; the analysis when this step is to the left is symmetric. There are two cases (refer to Fig. 3).

Case 1) If the previous step p_{i-1} to p_i is in the *same* direction as the current step p_i to p_{i+1}, then we cover the interval (p_i, p_{i+1}) of the step itself; there are now no sleeping robots in this interval that are awakened along this path.

Case 2) If the previous step p_{i-1} to p_i is in a *different* direction from the current step p_i to p_{i+1}, then we cover the interval of length r_i that touches p_i but is on the other side of p_i from p_{i+1}, that is, $(p_i, p_i - r_i)$. By the greedy property, we know that the interior of this interval contains no sleeping robots.

We now explain why the intervals that we cover are nonoverlapping. An important property of the greedy heuristic is that whenever step p_j to p_{j+1} is to the right (left), the point p_{j+1} is farther to the right (left) than points $p_1, p_2, \ldots, p_j$. We next analyze the previous cases in light of this observation.

Case 1) We cover (p_i, p_{i+1}). The interval (p_i, p_{i+1}) is new territory, and therefore could not have been covered by robots to the right of point p_i; similarly, the interval could not have been covered by a robot left of p_i because covered intervals have no sleeping robots in them, and p_i contained a sleeping robot until the previous step.

Case 2) We cover $(p_i, p_i - r_i)$. Since point p_i was the farthest point left in the previous step, we can show that interval $(p_i, p_i - r_i)$ was not previously covered by using similar reasoning as in Case 1.

Finally, we show that the maximum length of all charging is at most $2D$, where D is the diameter of the point set. Consider the last time that the path changes direction. This last segment has length at most D, and therefore can charge at most an extra D on one side of the trodden area. 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 theorem, which shows 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 Euclidean instances. Combined with our lower bounds, we get a *tight* analysis of the approximation factor for greedy in all dimensions $d \geq 1$.

Theorem 4: The greedy strategy, with claims, is an $O((\log n)^{1-1/d})$ approximation for the FTP on n points in d dimensions, for $d \geq 2$. The approximation bound holds, regardless of whether or not the strategy uses refresh or delayed target choice.

We give the proof below. We also show in *Theorem 5* that the greedy strategy can be *implemented* efficiently. Note the difference between *Theorems 4* and *5*. *Theorem 4* provides a bound on the awakening time (the height of the awakening tree), whereas *Theorem 5* provides a bound on the time to compute the awakening tree; that is, *Theorem 4* measures robot movement, whereas *Theorem 5* measures computing time. Since robots compute quickly but move slowly, the time to follow the awakening tree dominates the time to compute it.

Theorem 5: For any fixed $\epsilon > 0$, one can compute the greedy awakening tree using claims, refresh, and delayed target choice for n points in the plane in time $O(n^{1+\epsilon})$; in any higher dimension d, the time bound is $O(n^{2-2/(\lceil d/2 \rceil+1)+\epsilon})$. The same time bounds hold in the case in which the greedy strategy is applied with claims but without refresh and without delayed target choice.

Proof: Consider first the case in which the strategy uses refresh and delayed target choice. Then, at each event when a robot is awakened, we determine the next event by computing the minimum distance $d(p, q)$ between an awake robot (which resides at one of the original input points p, since the target choice is delayed) and a sleeping robot at some point q; the next event, then, will correspond to the awake robot at p going to awaken the sleeping robot at q. Thus, the next event is determined by the bichromatic closest pair between the "red" points (where awake robots are) and the "blue" points (where sleeping robots are). The bichromatic closest pair must be maintained *dynamically*, with insertions of "red" points and deletions of "blue" points, since each event results in a color change of some point. We apply the dynamic bichromatic closest pair data structure of Eppstein [9], who shows that the bichromatic closest pair can be maintained efficiently, with an amortized cost of $O(T(n) \log n)$ per insertion and $O(T(n) \log^2 n)$ per deletion, where $T(n)$ is the time required for performing nearest-neighbor queries or insertions/deletions in a nearest-neighbor data structure. Using the dynamic nearest-neighbor data structure of Agarwal and Matoušek [1], we obtain the claimed time bounds.

In the case that the greedy strategy is applied with claims but without refresh and without delayed target choice, we must determine, when a robot is awakened at point q, where the two robots now at point q will go next. This amounts to performing two nearest-neighbor queries. First, we query with the point q within the set of all unclaimed sleeping robots, finding the closest one, p; then, we delete p from the set of unclaimed robots and query again with the point q to determine the target for the other robot at q. The nearest-neighbor query data structure of [1] implies the claimed result. ■

Remark: In the L_1 or L_∞ metric, the time bound becomes $O(n \log^{O(1)} n)$ for any fixed dimension d, since we can apply orthogonal range-search data structures to perform nearest-neighbor queries. Also, we can use dynamic $(1+\epsilon)$-*approximate* nearest-neighbor query data structures (see, e.g., [7]) to construct an (approximate) greedy awakening tree in time $O(n \log n)$, using $O(n)$ space, for any $\epsilon > 0$ and any dimension $d \geq 2$.

Proof of Theorem 4: We concentrate on proving the approximation bound for the 2-D case of *Theorem 4*; 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 show that all paths in the awakening tree from the root to a vertex with out-degree at most 1 differ by at most an additive D distance. Thus, if we can provide a bound on the length of one of these paths, we have a bound on the longest path.

Claim 6: Consider the shallowest vertex with out-degree at most 1 and the deepest leaf in the awakening tree. The difference in depth between these two nodes is at most D, the diameter of the set of robots.

Proof: A node has an out-degree of at most 1 because all remaining sleeping robots have already been claimed (or, alternatively, because using this robot does not improve the makespan). Therefore, all robots that are still sleeping robots will be awakened at most D steps later. ■

Because the awakening tree has fewer than n leaves, we obtain the following corollary.

Corollary 7: There exists a path $P = (p_0, p_1, \ldots, p_K)$ from a root to a vertex with out-degree ≤ 1, where there are $K \leq \log n$ edges of lengths $r_i = \mathrm{dist}(p_i, p_{i+1})$.

We call such a path P a *maximal path*. At the time t when the last vertex p_K is reached, all of the remaining sleeping robots will be awakened by other branches of the awakening tree. The makespan, therefore, is at most $t + D \leq t + 2R$. We will show that $t = O(\sqrt{\log n} \cdot t^*)$, implying that the makespan is $O(\sqrt{\log n} \cdot t^*)$, where we recall that t^* is the optimal makespan.

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

Claim 8: No point $p_{i+1}, \ldots, p_K$ lies 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 4* 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 the inner circles $c_0, \ldots, c_{K-1}$ are disjoint (for the inner circles, this will hold whenever two circles differ substantially on radius, as we will show later on). The proof is based on the following lemma.

Lemma 9: The combined area covered by the (inner or outer) circles is $O(R^2)$, that is

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

Before proving *Lemma 9*, we show how to use this lemma to conclude the proof of *Theorem 4*. The length L of the path P is simply $L = \sum_{i=0}^{K-1} r_i$. By the Cauchy–Schwartz inequality,[2] the fact that $K \leq \log n$, and *Lemma 9*, we obtain

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

Since this result holds for any maximal path in the awakening tree, the approximation bound follows. This concludes the proof of *Theorem 4*. ■

Proof of Lemma 9: 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 10: Any pair of inner circles belonging to the same class are disjoint.

[2]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 $\mathbf{s} = (1, 1, \ldots, 1)$.

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 8*, since p_k is reached after p_j along the branch P of the awakening tree, $\operatorname{dist}(p_j, p_k) > r_j$. Since C_j and C_k belong to the same circle class, we know that $r_k < 2r_j$, implying that the radius of c_k, $(1/3)r_k$, is less than $(2/3)r_j$. Therefore, since $\operatorname{dist}(p_j, p_k) \geq r_j$, 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 the following.

Claim 11: 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 ℓ_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 $A_i/(2^i/3)^2\pi$ circles, since each inner circle of class i has radius at most $(2^i/3)$. 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 $9A_i/2^{i+1}\pi$. Similarly, we have at most $A_i/(2^{i-1}/3)^2\pi$ circles, since the (nonoverlapping) inner circles of class i each have radius at least $(2^{i-1}/3)$. 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 $9A_i/2^{i-2}\pi$. ■

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. Circles will overlap through the process of building the path P, but we need to distinguish, for the case of inner circles, those that do not overlap with any other inner circle (which we will use to claim that we are covering a part of the domain that has not been covered before) from the ones that do have overlaps. We need the following claim.

Claim 12: 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 of P that corresponds to edges whose inner circles do cover new area.

Proof: It suffices to consider the worst case. All areas covered by class-i inner circles is covered by class-j circles, for $j > i$. In class i, the average amount that we must walk along P in order to cover one unit of area is between $9/2^{i+1}\pi$ and $9/2^{i-2}\pi$. Thus, suppose that a unit of area is covered first in class i, where the average cost is between $9/2^{i+1}\pi$ and $9/2^{i-2}\pi$. The cost of covering this area in larger classes is a geometric series summing 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 12*, 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 9* follows. ■

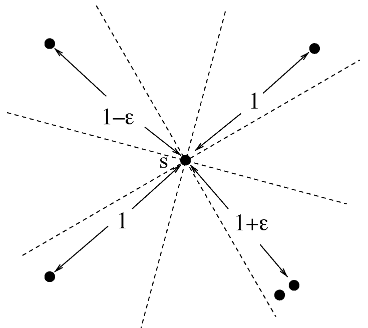

Fig. 4. Example in which greedy is suboptimal. Greedy initially sends the robot at s to the northwest corner, while a better strategy (maximizing the "bang for the buck," defined in the text) sends the robot at s initially to the southeast corner, where two nearby robots are awakened at essentially the same time, leading to a better overall wakeup strategy (makespan 3 versus $1 + 2\sqrt{2}$).

The proof for the d-dimensional case of *Theorem 4* is similar, considering a volume-covering argument with d-dimensional balls and a corresponding set of claims and lemmas as above.

B. 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 Fig. 4). In this section, we examine alternative strategies in an attempt to overcome this weakness.

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[3] 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° 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, at most k options face a robot once it is activated. Which sector should be selected? Once the sector is selected, the robot heads for the closest sleeping robot it has sensed in that sector. A greedy strategy that uses sensor sectors will select the sector whose closest robot is the closest.

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 not-claimed sleeping robot in the sector. In Fig. 4, the bang for the buck strategy is significantly better than greedy; Fig. 5 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

[3]Joint with HRL Labs. [Online]. Available: http://www.hrl.com/

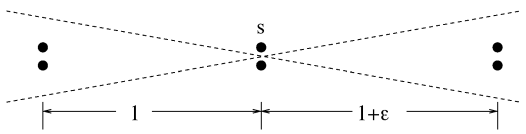

Fig. 5. Example in which both greedy and bang for the buck strategies are suboptimal. Both strategies use the awake robot at s to awaken the nearby robot and then send both robots to the left, while an optimal strategy would send one robot to the left, one to the right (makespan 1 versus 3).

sector and chooses its next target to be the closest sleeping robot in that sector.

In the case where the closest sleeping robot is already claimed by another robot, a new sector is randomly selected and the closest sleeping robot is chosen. If this situation occurs in all k sectors, once a new sector is selected, the robot targets the second-closest sleeping robot for 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 is chosen to be the closest sleeping robot in a cone centered on a vector to the west, while the target for robot B to awaken next is selected from a cone centered on a vector to the east. More specifically, at an awakening event, the robot A that does the awakening selects its next target to be the closest sleeping robot from the *opposite cone*, the cone consisting of c consecutive sectors centered on the sector $k/2$, opposite to its heading (sector 0). The newly awakened robot B selects its next target to be the closest sleeping robot in the c-sector cone centered on A's heading (sector 0) at the moment of its awakening. If there are no unclaimed sleeping robots within a cone, the cone is successively enlarged by incrementing the cone width c.

III. Experiments

A. 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 nongeometric datasets. We investigated five classes of geometric datasets.

1) Uniform: The n points that represent the sleeping 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 sleeping 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[4] that have data of type EUC_2D (68 instances).

Since our nongreedy strategies are specified geometrically, they were applied only to the geometric data.[5]

The greedy strategies were applied to nongeometric datasets, including the following.

1) Star Metrics: $n-1$ sleeping 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. For some experiments, we chose the selection to be uniform between 1 and n, while for others, we picked the uniform selection on fixed ranges, e.g., in (100, 200). Distances are measured according to path length in the star. We consider stars having exactly one sleeping robot per leaf (case "1-1") and stars having potentially many sleeping 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: We selected instances from the TSPLIB, using the symmetric traveling salesman files, with data of type MATRIX (14 instances). These metrics are TSP instances that are derived from nongeometric problems.

Adapting our algorithms to nongeometric data, we needed to address the problem of *stopping* a robot that is in motion along an edge of the network toward 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, 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 (i.e., the makespan, the time until all robots are awake); 2) *total distance* traveled by all robots; and 3) *average distance* traveled by robots that do any traveling. 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.

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 II-A

[4][Online] Available: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[5]While our nongreedy strategies might be generalized to nongeometric datasets, this has not been a part of our experiments.

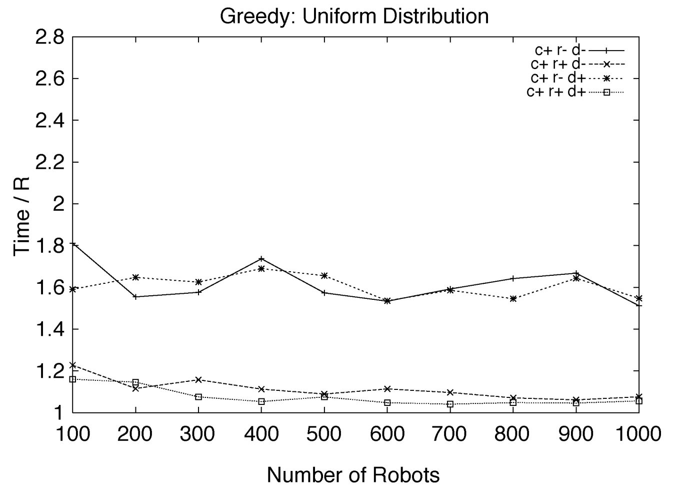

Fig. 6. Greedy on uniform distributions. Choices of parameters for claims (c), refresh (r), and delayed target choice (d). A "+" ("−") indicates the parameter is set to *true* (*false*). Standard deviations for each case: .49 (+c); .14 (+c+r); .47 (+c+d); .09 (+c+r+d). Using refresh is seen clearly to be desirable; among those using refresh, using the delayed target choice is somewhat better.

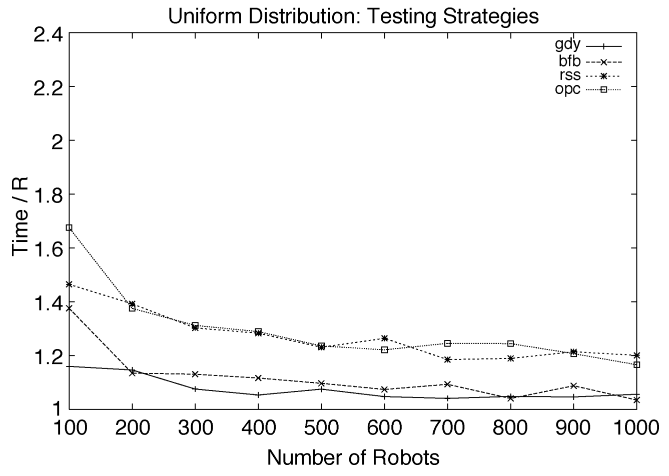

Fig. 7. Greedy (gdy), bang for the buck (bfb), random sector selection (rss), and opposite cone (opc) tested on uniformly distributed swarms. Standard deviations for each case: 0.09 (gdy), 0.16 (bfb), 0.19 (rss), 0.22 (opc). The four strategies give a constant approximation to the lower bound with values between 1 and 1.5. Greedy gives the better approximation.

Claims: Our implementation assumes the claims with local communication model described earlier. If this parameter is set to *true*, when a robot is awakened, it and the robot that awakened it select distinct sleeping robots as targets; otherwise, they may select the same sleeping robot as a 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 sleeping 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 (which requires a robot model in which look-ahead is possible) allows a robot to stay on the same spot when its 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 awakened robots and their selected targets. The robot accumulates units of distance that are used toward its target when the amount of units accumulated plus the step distance allows it to reach that target.

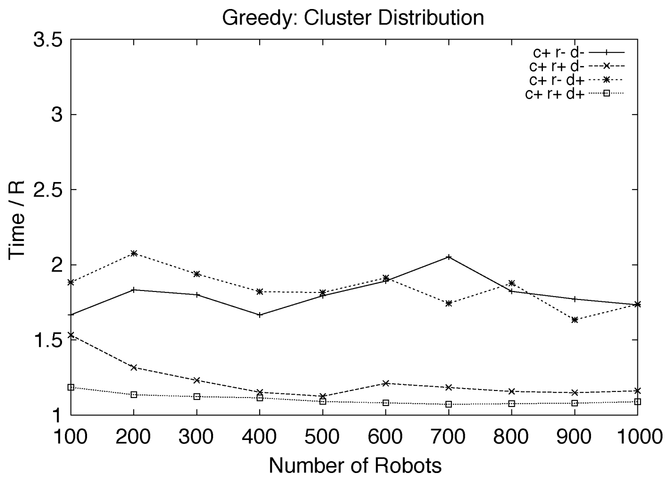

Fig. 8. Greedy on cluster distributions; different combinations of choices of parameters for claims (c), refresh (r), and delayed target choice (d). A "+" ("−") indicates the parameter is set to *true* (*false*). Standard deviations for each case: 0.62 (+c), 0.28 (+c+r), 0.62 (+c+d), 0.15 (+c+r+d). As 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).

B. 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 $n \in \{100, 200, 300, \ldots, 1000\}$.

The experiments on datasets from the TSPLIB (EUC_2D or MATRIX) were done once per dataset; one sleeping 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, and 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 significant; without it, the robots travel in groups rather than spreading out. (In comparing our claims with local communication with our prior implementation of claims with global communication, we found little difference in performance; compare with [18].) There is an advantage in using the refresh option. While

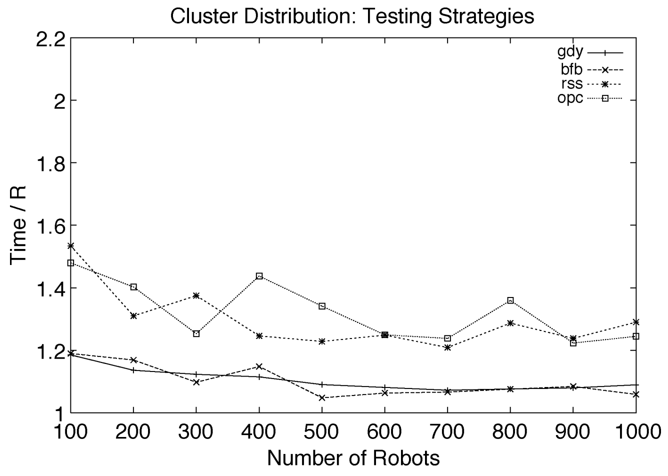

Fig. 9. Strategy comparison on cluster data. Standard deviations for each case: 0.15 (gdy), 0.16 (bfb), 0.45 (rss), 0.44 (opc). Greedy generally outperforms the other strategies, with bang for the buck holding a close second. The approximation factors are essentially constant, with similar values to the uniform datasets.

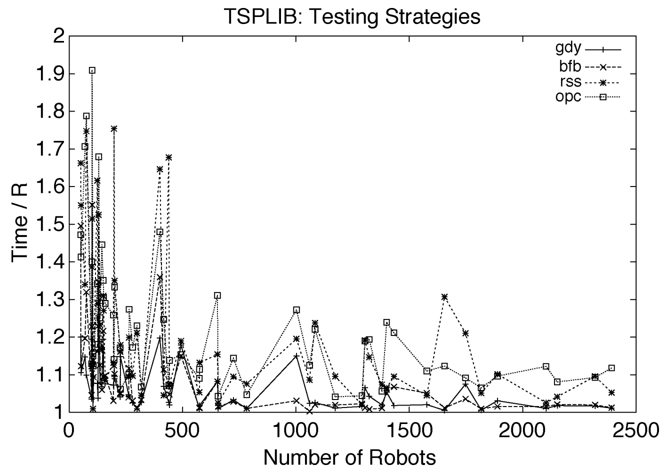

Fig. 10. Strategy comparison on TSPLIB EUC_2D data.

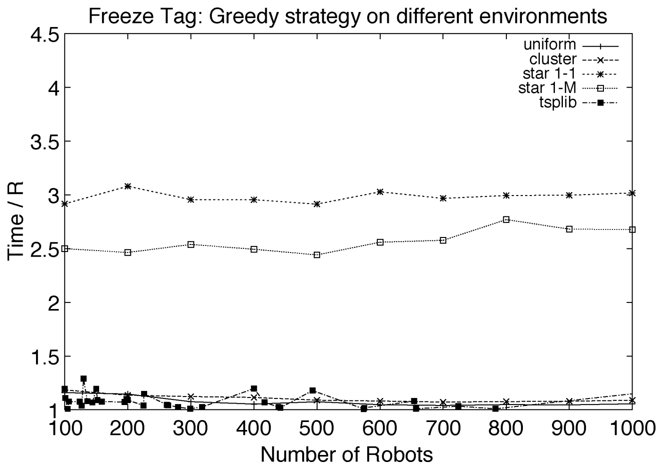

Fig. 11. Comparing greedy on five different datasets. Standard deviations for each case: 0.09 (unf), 0.15 (clu), 1.26 (stars). Note that the approximation factor stays relatively flat (constant) with swarm size n. While the factor is between 1.0 and 1.3 for the uniform and cluster cases and TSPLIB (EUC_2D), 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 (one robot per leaf). It is interesting to note that the TSPLIB results are indistinguishable from the uniform and cluster data results.

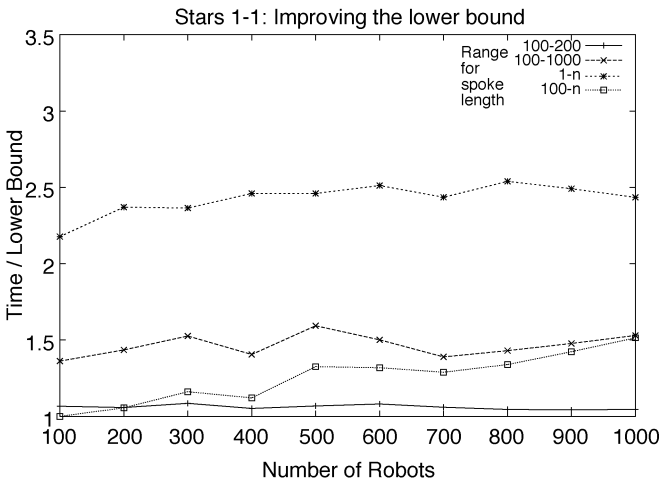

Fig. 12. Comparing greedy on star 1-1 datasets, using the improved lower bound of $2L_{\min}(\lceil\log(n+1)\rceil - 1) + L_{\max}$. Each curve corresponds to a randomly generated dataset having spoke lengths uniformly generated in the indicated interval. Note that for spoke lengths that are generated in the length interval (100 200), the approximation factor is essentially 1.

less substantial, the delayed target choice is also advantageous (though it requires a robot model in which look-ahead is possible). Fig. 6 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 solid heuristic, most often outperforming the other strategies in our comparisons. While other strategies achieve gains during the initial iterations of a simulation by reaching robots at farthest distances, once those robots are awaken there are no more gains for those strategies to achieve, thus making them, from that point on less efficient than greedy (see Figs. 7–10). As the size of the swarm increases, the approximation ratios tend to stay about the same or decrease; we suspect this decrease 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 higher (between 2 and 3), but this is because R is a poor lower bound in the case of stars. In order to support this explanation, we computed an alternative lower bound specifically for star metrics having one robot per leaf. (The lower bound is $2L_{\min}(\lceil\log(n+1)\rfloor - 1) + L_{\max}$, where $L_{\min}$ ($L_{\max}$) is the length of the shortest (longest) spoke of the star. More complex lower bounds can be similarly derived for the case of stars with many robots per leaf.) Fig. 11 compares greedy on different datasets, showing that performance is better for the uniformly distributed and the cluster cases. Fig. 12 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 (Figs. 13 and 14) showing the percentage of wins for each strategy, and, finally, report in Fig. 15 the results for greedy on the nongeometric 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

Fig. 13. Top: 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 two out of three runs. Bottom: 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 led over the second-place strategy.

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

Fig. 14. Comparing strategies on uniform datasets. Top: 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 two out of three runs. Bottom: 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 led over the second-place strategy.

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

Fig. 15. Results of greedy strategy for TSPLIB MATRIX (nongeometric) 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).

IV. Conclusion

We have performed theoretical and experimental analysis of variants of the greedy strategy and other heuristics for the FTP. Our theoretical analysis is tight, showing that greedy performs within a factor $\Theta((\log n)^{1-1/d})$ of optimal. Our experiments show, however, that greedy (and other heuristics) perform 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 the best possible approximation algorithms for general metric spaces.

Note Added in Proof: A new $O(\sqrt{\log n})$-approximation algorithm has just been announced by Konemann *et al.* [24].

Acknowledgment

The authors would like to thank D. Gage for discussions motivating this research and G. Bachelis for useful discussions on the upper-bound analysis of greedy. They would also like thank the referees for many helpful suggestions that improved the paper.

References

[1] P. K. Agarwal and J. Matoušek, "Dynamic half-space range reporting and its applications," *Algorithmica*, vol. 13, pp. 325–345, 1995.
[2] S. Albers and M. R. Henzinger, "Exploring unknown environments," in *Proc. 29th Annu. ACM Symp. Theory of Computing*, 1997, pp. 416–425.
[3] S. Albers, K. Kursawe, and S. Schuierer, "Exploring unknown environments with obstacles," in *Proc. Symp. Discrete Algorithms*, 1999, pp. 842–843.
[4] 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 Symp. Discrete Algorithms*, 2002, pp. 568–577.
[5] E. M. Arkin, M. A. Bender, D. Ge, S. He, and J. S. B. Mitchell, "Improved approximation algorithms for the freeze-tag problem," in *Proc. 15th ACM Symp. Parallelism in Algorithms and Architecture*, 2003, pp. 295–303.
[6] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber, "Multicasting in heterogeneous networks," in *Proc. 30th Annu. ACM Symp. Theory of Computing*, May 23–26, 1998, pp. 448–453.
[7] S. N. Bespamyatnikh, "Dynamic algorithms for approximate neighbor searching," in *Proc. 8th Canadian Conf. Comput. Geom.*, 1996, pp. 252–257.
[8] A. M. Bruckstein, C. L. Mallows, and I. A. Wagner, "Probabilistic pursuits on the grid," *Amer. Math. Monthly*, vol. 104, no. 4, pp. 323–343, 1997.
[9] D. Eppstein, "Dynamic Euclidean minimum spanning trees and extrema of binary functions," *Discr. Comput. Geom.*, vol. 13, pp. 111–122, 1995.
[10] D. Gage, "Minimum-resource distributed navigation and mapping," in *Proc. SPIE Mobile Robots XV*, vol. 4195, Boston, MA, Nov. 2000, pp. 96–103.
[11] S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *NETWORKS*, vol. 18, pp. 319–349, 1988.
[12] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring an unknown cellular environment," in *Abstracts 16th Eur. Workshop Comput. Geom.*, 2000, pp. 140–143.
[13] R. Ravi, "Rapid rumor ramification: Approximating the minimum broadcast time," in *Proc. 35th Annu. Symp. Foundations of Computer Science*, S. Goldwasser, Ed., Los Alamitos, CA, Nov. 1994, pp. 202–213.
[14] R. V. Sole, E. Bonabeau, J. Delgado, P. Fernández, and J. Marín, "Pattern formation and optimization in army ant raids," *Artif. Life*, vol. 6, pp. 219–227, 2001.
[15] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," *J. Robot. Syst.*, vol. 13, no. 3, pp. 127–139, 1996.
[16] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots—formation and agreement problems," in *Proc. 3rd Int. Colloq. Structural Information and Communication Complexity*, Siena, Italy, 1996, pp. 313–330.
[17] ——, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1347–1363, 1999.
[18] M. Sztainberg, E. M. Arkin, M. A. Bender, and J. S. B. Mitchell, "Analysis of heuristics for the freeze-tag problem," in *Proc. 8th Scand. Workshop Algorithm Theory*, 2002, vol. 2368, pp. 270–279.

[19] I. Wagner and A. Bruckstein, "Cooperative cleaners: A study in ant robotics," Technion, Haifa, Israel, Tech. Rep. CIS9512, 1995.

[20] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 918–933, Oct. 1999.

[21] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Efficiently searching a graph by a smell-oriented vertex process," *Ann. Math. Artif. Intell.*, vol. 24, no. 1–4, pp. 211–223, 1998.

[22] ——, "MAC vs. PC—determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains," *Int. J. Robot. Res.*, vol. 19, no. 1, pp. 12–31, 2000.

[23] A. Paulraj, V. Roychowdhury, and C. D. Schaper, Eds., *Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath*. Amsterdam, The Netherlands: Kluwer, 1997, pp. 289–308.

[24] J. Konemann, A. Levin, and A. Sinha, "Approximating the degree-bounded minimum diameter spanning tree problem," in *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, ser. Lecture Notes in Computer Science. New York: Springer-Verlag, 2003, vol. 2764, pp. 109–121.

Marcelo O. Sztainberg received the B.S. degree in computer science and mathematics from Wayne State University, Detroit, MI, in 1998, and the M.S. and Ph.D. degrees in applied mathematics and statistics from the State University of New York, Stony Brook, in 1999 and 2003, respectively.

He is an Assistant Professor of Computer Science with Northeastern Illinois University, Chicago. His research interests include swarm robotics, analysis of algorithms, computational geometry, and nanotechnology.

Esther M. Arkin received the B.S. degree in mathematics from Tel-Aviv University, Tel Aviv, Israel, in 1981, and the M.S. and Ph.D. degrees in operations research from Stanford University, Stanford, CA, in 1983 and 1986, respectively.

After five years on the faculty of the School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, she joined the faculty at the State University of New York, Stony Brook in 1991, where she is now Professor of Applied Mathematics and Statistics and a Research Professor of Computer Science. Her research interests are the design and analysis of algorithms for a variety of fields, including network optimization, computational geometry, graph theory, and scheduling.

Michael A. Bender received the B.A. degree in applied mathematics from Harvard University, Cambridge, MA, in 1992, the D.E.A. degree in computer science from the Ecole Normale Superieure de Lyon, Lyon, France, in 1993, and the Ph.D. degree from Harvard University in 1998.

He is an Assistant Professor of Computer Science with the State University of New York, Stony Brook. His research interests include analysis of algorithms, data structures, scheduling, parallel computing, cache- and I/O-efficient computing, and robot algorithms. He has coauthored over 60 articles on these and other topics in computer science.

Joseph S. B. Mitchell received the B.S. degree in physics and applied mathematics and the M.S. degree in mathematics from Carnegie-Mellon University, Pittsburgh, PA, in 1981, and the Ph.D. degree in operations research from Stanford University, Stanford, CA, in 1986.

He was with Hughes Research Laboratories from 1981 to 1986, and then on the faculty of Cornell University, Ithaca, NY, from 1986 to 1991. He now serves as a Professor of Applied Mathematics and Statistics and Research Professor of Computer Science at the State University of New York, Stony Brook. His primary research area is computational geometry, applied to problems in computer graphics, visualization, air traffic management, manufacturing, and geographic information systems.