The Online Freeze-tag Problem

Mikael Hammar[1], Bengt J. Nilsson[2], and Mia Persson[2]

[1] Apptus Technologies AB, IDEON, SE-223 70 Lund, Sweden
mikael.hammar@apptus.com
[2] School of Technology and Society, Malmö University, SE-205 06 Malmö, Sweden
Bengt.Nilsson@ts.mah.se, mia@cs.lth.se

Abstract. We consider the following problem from swarm robotics: given one or more "awake" robots in some metric space M, wake up a set of "asleep" robots. A robot awakens a sleeping robot by moving to the sleeping robot's position. When a robot awakens, it is available to assist in awakening other slumbering robots. We investigate offline and online versions of this problem and give a 2-competitive strategy and a lower bound of 2 in the case when M is discrete and the objective is to minimize the total movement cost. We also study the case when M is continuous and show a lower bound of 7/3 when the objective is to minimize the time when the last robot awakens.

1 Introduction

The *Freeze-tag problem* has received some attention recently [1, 2, 7]: given a set of n robots located at points in some metric space. Initially, there is one or more *awake* or *active* robots and the other robots are *asleep*, i.e., in a *stand-by mode*. The objective is for the active robots to awaken the sleeping ones. An active robot is able to awaken a sleeping robot just by touching it. Once awake, this new robot is available to assist in awakening other sleeping robots. This problem was dubbed the Freeze-tag problem by the authors of [1, 2, 7].

In this paper, we consider online versions of this problem where the sleeping robots, also denoted as *requests*, occur in an online fashion, i.e., neither the total number of requests nor their exact location in space are known in advance. Therefore, decisions, i.e., which robots to move in order to wake up sleeping robots, has to be made without any knowledge about future requests. We consider the following two online problems. The *online step dependent Freeze-tag problem* (online $SDFT(k)$ for short) has k awake robots from the start and a request is released only when the previous request has been activated or served and the objective is to minimize the total distance travelled by all the robots. The *online time dependent Freeze-tag problem* (online $TDFT$ for short), closely models the Freeze-tag problem defined in [1] but in an online setting. The requests are released independently of how many of them have already been served and our objective is to minimize the makespan.

The Freeze-tag problem has many applications such as data distribution, network design, broadcasting, routing and scheduling [1]. Another feature is

that *SDFT*(k) and *TDFT* exhibit some elements of the k-server problem and the Traveling Salesman Problem, respectively. In the well studied k-server problem (see e.g. [4, 5]) one aims to online serve a set of requests, positioned on points in some metric space M, by k mobile servers and simultaneously minimize the total movement cost. Here a request is a point in M and the servers are located on points of M. A request r is said to be served if one of the servers lies on r. Manasse *et al.*[5] have shown that k is a lower bound on the competitive ratio of any deterministic k-server algorithm in any metric space with at least $k+1$ points, whereas the work function algorithm for the k-server problem has been shown to have a competitive ratio of at most $(2k-1)$[4]. The online step dependent Freeze-tag problem closely models the online k-server problem but with the difference that now the number of servers, i.e., active robots, is not constant; a new server appears for each served request. In this paper, we provide a 2-competitive online algorithm for this variant of the k-server problem.

The online time dependent Freeze-tag problem can be viewed as a parallel online version of the Traveling Salesman Problem, in which the cities of the TSP instance correspond to the asleep robots initial locations in time dependent Freeze-tag, and the objective is to visit all cities as rapidly as possible. Furthermore, in *TDFT* there are many salesmen working in parallel since whenever a salesman visit a city, he recruits a new salesman to help visit other cities. The time dependent Freeze-tag problem was first introduced by Arkin *et al.*[1]. They showed that in the offline case, minimizing the time when the last robot awakens, even simple versions of the problem (e.g. in star metrics) are NP-complete and they provide a PTAS for geometric instances on a set of points in any constant dimension. Sztainberg *et al.*[7] further investigate offline Freeze-tag and prove that the greedy strategy gives a tight approximation bound for the case of points in the plane and that greedy yields a $\Theta((\log n)^{1-1/\delta})$-approximation for n points in R^{δ}. Arkin *et al.*[2] recently prove the NP-hardness and present an $O(1)$-approximation for offline Freeze-tag in unweighted graphs, in which there is one asleep robot at each node. They further generalize to the case when there are multiple robots at each node and edges are unweighted and they obtain a $\Theta(\sqrt{\log n})$-approximation. In the case of weighted edges for this problem, they provide an $O((L/d)\log n + 1)$-approximation, where L is the edge of heaviest weight and d is the diameter of the graph.

1.1 Our Results

In this paper we show that offline *SDFT*(k) can be solved in polynomial time for all possible values of the parameter k. We further give a 2-competitive algorithm in the online case and also prove a lower bound of 2 on the competitive ratio for all k. For online *TDFT* we prove a lower bound of 7/3 on the competitive ratio.

Our paper is organized as follows. In section 2 we give some basic definitions and also formally define the different variants of the Freeze-tag problems. In section 3 we consider the offline version of *SDFT*(k) and prove that this problem can be solved in polynomial time. In section 4 we present a strategy yielding a competitive ratio of 2 for online *SDFT*(k) and then we further prove that this is

in fact optimal. In section 5 we prove a lower bound of 7/3 on the competitive ratio for the *TDFT* problem.

2 Preliminaries

We begin with some notation and basic definitions.

Given a metric space M, the robots are represented as points in M. A robot can be in two different states, the *awake* state and the *asleep* state. We also call sleeping robots *requests* and say that a request is *served* when a sleeping robot is awakened by an active robot. An active robot is also denoted as a *server*. The *release time* of robot r specifies the first point in time when r can be awakened and also the first point in time when the awake robots become aware of r.

The formal definition of the problems is as follows.

Definition 1. *In the online Step Dependent Freeze-tag problem,* (SDFT(k) *for short*)*, one is given a set of k initially awake robots located on points in some discrete metric space M and the objective is to serve all requests in such a way that the total movement cost is minimized. A new request is released only when the previous request has been served.*

In the online Time Dependent Freeze-tag problem, (TDFT *for short*)*, one is given one initially awake robot, located on a point in some continuous metric space L and the objective is to serve all requests in such a way that the makespan, i.e., the time when the last robot is awakened, is minimized. An awake robot can move with at most unit speed and associated to each robot is a release time.*

Note that in the *TDFT* problem a moving robot is allowed to reconsider its choice, i.e., aborting its motion when a new request occur. This is not possible in the *SDFT*(k) problem since a new request is released only when the previous request has been served.

The performance of deterministic online algorithms is measured in comparison with the optimal offline algorithm, denoted by *OPT*, using the standard competitive ratio, see e.g. [3]. It is assumed that *OPT* knows the entire input sequence, i.e., the total number of requests and their locations in M, and can hence achieve a lower cost. Furthermore, an online algorithm A is c-competitive for a constant c, if for all input sequences σ, the following holds: $A(\sigma) \leq cOPT(\sigma)$. The infimum of all such values c over all request sequences σ is called the competitive ratio of A. The input to an online algorithm is constructed by an *adversary*. For deterministic online algorithms, a *cruel adversary* knows exactly what the online algorithm's response will be to each input element and this adversary pays the optimal offline cost. A different kind of adversary is the *adaptive-online adversary*, who must serve each request it generates before the randomized online algorithm serves the request and this adversary knows its own strategy for generating requests as well as the description of the online algorithm and all its action taken thus far (see e.g. [3] for a more comprehensive survey on adversaries).

We refer to an algorithm A as being *lazy* if (1) A moves its robot along the shortest path to a sleeping robot and (2) A moves only one robot in order to serve a request.

The following lemma proves that any non-lazy online algorithm for *SDFT*(k) can be replaced by a lazy online algorithm without increasing the algorithms total movement cost.

Lemma 1. *Any online algorithm for the SDFT(k) problem can be replaced by a lazy online algorithm without increasing the total path travelled by the robots.*

Proof. Consider a feasible solution, i.e., a sequence $\sigma = (r_1, \ldots, r_n)$ of serving robots, generated by a non-lazy algorithm A for *SDFT*(k). We show how to construct a new feasible solution by replacing all non-lazy movements by lazy movements without increasing the total distance travelled by the robots. The construction goes as follows. First, let us assume that A moves robot $r_i \in \sigma$ in order to wake up the sleeping robot $r(j)$ and furthermore, assume that A makes its first non-lazy movement in this step, denoted as the j:th step. This non-lazy movement is clearly (at least) one of the following kind: (1) A does not move along the shortest path to serve $r(j)$, or (2) A moves more than one robot to serve $r(j)$. Now replace this non-lazy movement by a lazy movement by moving only r_i along the shortest path to robot $r(j)$ and no further. It is clear that this replacement will not increase the total distance travelled by the robots up to step j. Repeat this process of exchanging non-lazy moves by lazy ones until all moves are lazy, using the same serving robot sequence, i.e., σ, as A. Since we only consider discrete metric spaces, the exchange of non-lazy moves to lazy ones will never result in a loss of cost. □

3 Offline Step Dependent Freeze-tag

In this section we consider the offline version of *SDFT*(k) and we show that this problem has a polynomial time solution by reducing it to maximum cardinality minimum weight matching on bipartite graphs, which is known to have polynomial time algorithms (see e.g. [6]).

We reduce *SDFT*(k) to maximum cardinality minimum weight matching on bipartite graphs. Let $R = (r_1, \ldots, r_n)$ be a set of n requests where r_j denotes the j:th request in the sequence, where j goes from 1 to n. Let $S_0 = (s_{0_1}, \ldots, s_{0_k})$ be a set of k initially awake robots. We further define two new sets $S = (s_1, \ldots, s_n)$ and $S' = (s'_1, \ldots, s'_n)$ such that $s_j = s'_j = r_j$. Thus, S and S' are copies of R. Now define a graph $G = (S_0 \cup S \cup S', R, E)$, whose vertices represent the set of requests and the set of servers, i.e., active robots that can be used to awaken the sleeping robots. Clearly, G is bipartite because the set of vertices can be partitioned into two sets, one set containing the vertices corresponding to the servers (set $S_0 \cup S \cup S'$) and the other containing the vertices corresponding to the requests (set R). Two vertices $(s_i, r_j) \in G$ (or $(s'_i, r_j) \in G$), with $1 \leq i \leq n$, are adjacent if and only if $i < j$. Also, for each $s_{0_i} \in G$, there are edges connecting s_{0_i} to every $r_j \in R$. Furthermore, an edge (s_{0_i}, r_j) (or (s_i, r_j) or (s'_i, r_j), respectively) is assigned weight equal to the distance between request r_j and server s_{0_i} (or s_i or s'_i, respectively). Note that if r_i and r_m, for $1 \leq m \leq n$, are two distinct requests with $i < m$, that happen to occur on the same point, then the corresponding

vertices in G are considered as being distinct, although the edges (s_i, r_m) and (s'_i, r_m) have zero weights. Furthermore, the edges (s_i, r_i) and (s'_i, r_i) does not exist for any $r_i \in R$. It suffices to prove that the bipartite graph G has a maximum cardinality minimum weight matching of size l if and only if the total movement cost for the robots in *SDFT*(k) is l. Let $\{e_1, \ldots, e_n\}$ be a set of edges in a maximum cardinality minimum weight matching on G. Since $|S_0| > 1$ and an edge $(s_i, s_j) \in E$ for each $i < j$ it follows that a maximium cardinality minimum weight matching covers all vertices in R exactly once, whereas a vertex in set $S_0 \cup S \cup S'$ is covered at most once. Furthermore, note that the set of edges $E \in G$ simulates the precedence constraints in *SDFT*(k), i.e., a sleeping robot cannot be used for awaking other robots before it has been awaken. Hence, $\{e_1, \ldots, e_n\}$ is a set of movements costs with minimum cost for the serving robots in *SDFT*(k). Conversely, let $\{l_1, \ldots, l_n\}$ represent a set of movements costs of minimum cost for the serving robots generated from the *SDFT*(k) execution process on graph G. Since there is only one active robot located on each vertex in $S_0 \cup S \cup S'$ in G and since each request is served exactly once it follows that $\{l_1, \ldots, l_n\}$ forms a maximum cardinality minimum weight matching in G. See Fig. 1 for an illustrating example of the reduction, where the dashed edges indicate an optimal solution of cost 7 for *SDFT*(k).

We have proved the following theorem:

Theorem 1. *The problem SDFT(k) can be solved in polynomial time.*

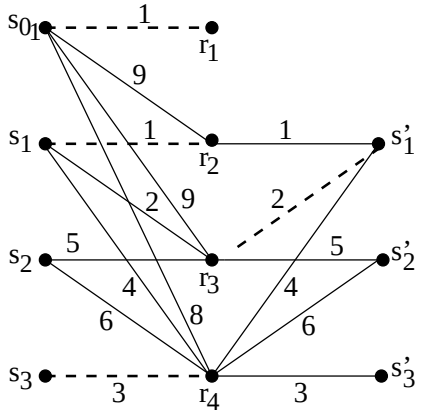

Fig. 1. Graph G for sequence of requests (r_1, r_2, r_3, r_4) with initially one active server s_0.

4 Online Step Dependent Freeze-tag

We now present a strategy, *Simple*, which achieves a competitive ratio of two for *SDFT*(k). This strategy works in arbitrary metric spaces and for all possible

values of the parameter k. *Simple* works as follows. For each request in the sequence, the request is served with the closest robot which afterwards returns to its original position. Hence, *Simple* is a non-lazy strategy.

Theorem 2. *Simple is 2-competitive.*

Proof. To prove our claim about a competitive ratio of two for our strategy, we just need to consider that for any request in the request sequence, the optimal strategy must move at least the minimum distance from a previous request. *Simple* moves exactly this distance in order to serve the request and then back to its original position, thus proving the theorem. □

By using Lemma 1 *Simple* can be made lazy without increasing its competitive ratio.

We proceed by proving that *Simple* is in fact optimal. We start by giving an easy argument to why two is a lower bound for $SDFT(k)$ when $k \geq 2$, i.e., there are at least two active robots initially.

Theorem 3. *There exist metric spaces for which the competitive ratio for the SDFT(k) is at least* 2*, when* $k \geq 2$*.*

Proof. Given a $k+1$ point unit-distance metric space M, i.e., M is a metric space with $k+1$ points such that for each pair of points p and q, distance(p, q)=1. Let A be any deterministic online algorithm for the $SDFT(k)$ problem. We show that there exists a request sequence σ such that $A(\sigma) \geq 2OPT(\sigma)$. By Lemma 1, we assume that A is lazy. For the construction of σ, let the initial configuration of A consist of k active robots, occupying k distinct points in M, i.e., initially there exist one point p in M not occupied by an active robot. Now, the move for the cruel adversary is to put request r_1 on point p. Then, assume that A serves r_1 with one of its robots positioned at say point q, with q occupied by one active robot r. Hence, this movement of r leaves an empty point at position q and the next move for the cruel adversary strategy is to put the second request r_2 on position q, forcing A to move a total distance of two in order to serve σ. Clearly, $OPT(\sigma) = 1$ since OPT can move some other active robot than robot r to serve r_1 and this concludes the proof. □

Next, we prove a lower bound holding for all possible values of the parameter k, i.e, this lower bound holds also in the case when $k = 1$. We begin by defining a few concepts. A metric space is a *tree metric* if the underlying metric graph structure is a tree. The tree metric has *unit cost* if all tree edges have weight one. The distance between two points is the sum of the edge weights on the tree path between the points. Tree metrics generalize such metric spaces as star metrics and line metrics. An online algorithm for $SDFT(k)$ in a tree metric is said to have the *nearness* property if it never serves a request using an active robot having some other active robot on the tree path to the request.

To show that we can restrict ourselves to consider only online algorithms having the nearness property, we first need to prove the following lemma.

Lemma 2. *For any online algorithm that solves SDFT(k) in tree metrics there is an online algorithm having the nearness property that solves the same instance with the same or better competitive ratio.*

Proof. Consider a request r that the algorithm serve with a robot r' without using nearness. This means that there is an active robot r'' on the tree path from r' to r and we can equivalently view the algorithm as moving r' to the position of r'' and let r'' serve the request r. However, this step is non-lazy and by Lemma 1 we can defer the movement of r' to the position of r'' to a later request, when it is needed. □

We are now in a position to prove our lower bound result.

Theorem 4. *There is a unit cost tree metric such that no randomized online algorithm for SDFT(k) has competitive ratio* $2-\epsilon$*, for any* $\epsilon > 0$*, against adaptive online adversaries.*

Proof. We let an adaptive online adversary construct a request sequence consisting of n requests in a unit cost tree metric where the tree initially has one or more active robots at a designated node that we denote the root r, i.e., the tree is a rooted tree. Each node in the tree has degree n and the distance from the root to each leaf node is n.

The adversary now places requests at different children of the root until the online algorithm has used all robots at the root to serve these requests. Note that by Lemma 2 we can assume that the online algorithm has the nearness property.

We denote a point that previously had active robots but now does not by a *hole* and a point that has a single active robot as a *single point*. Hence, once the initially active root robots all have served the requests at the children we have a hole at the root and an even number of active robots, two at each point having active robots. The adversary continues constructing requests in accordance with the following simple scheme S.

1. If the tree has a hole, then a request is placed at the hole.
2. If the tree has a single point, then a request is placed at a child of the single point being the root of a subtree having no active robots.

A simple induction proves that after the online algorithm has served each request, then the tree will either contain an even number of robots and a hole or an odd number of robots with one single point. All other points having active robots will have two active robots positioned at the same point. Again we rely on the fact that the online algorithm has the nearness property.

Let us divide the subsequent work of the online algorithm into stages. A *stage* begins with the online algorithm A and an adversarial online algorithm A^* both having a hole at the root of the tree and ends after a number of requests have been served when A and A^* again reach the same configuration, i.e., both have their hole at the root. The number of children of the root that have active robots we henceforth call the *r-degree*. Consider now an arbitrary stage i with

r-degree $d_i \geq 2$. The adversary places a request at the root and A serves that request using an active robot at one of the children of the root, say p. A^* on the other hand serves that request using an active robot at some other child q of the root. The stage now continues with the adversary placing requests according to the simple scheme S to be served by both A and A^*. If A and A^* reach the same configuration, i.e., both have a single point at q, then the two subsequent requests will be handled by A^* so that it has a single point at the root. Without loss of generality we can assume that A does the same, thus one more request will generate a hole at the root ending the stage and starting the next stage. Figure 2 gives an example of a stage. The label of an edge is the sum of the movements cost along this edge during the stage. Note that there are always two active robots on each node (except for the root of the tree which has no active robot) after a stage.

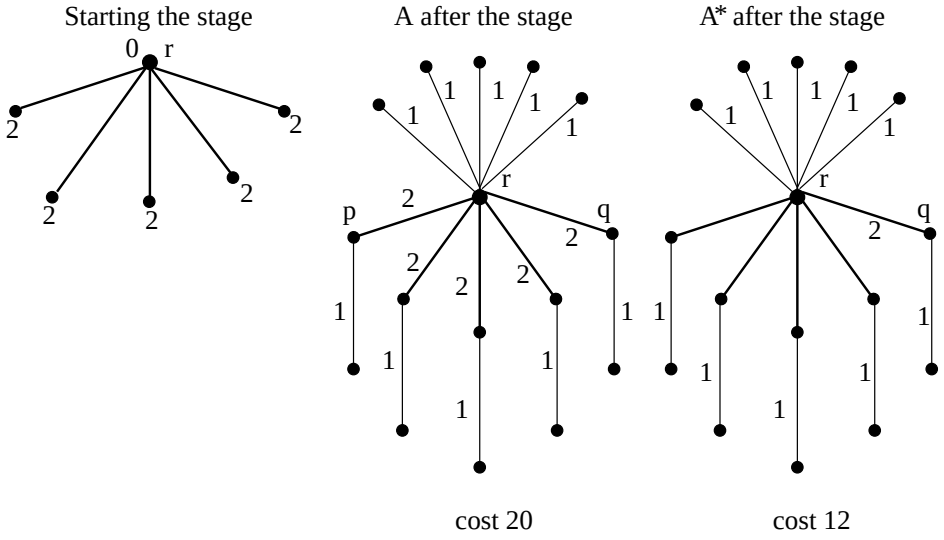

Fig. 2. Example illustrating the proof of Theorem 4.

Let us analyze the expected cost of A with respect to the expected cost of A^* during stage i. The adversary places a request at the root, which is served by A and A^*. Note that, independently of the probability distribution that A uses, the probability that A and A^* serve that request using the same active robot is $\frac{1}{d_i}$ if A^* chooses the robot to use with uniform distribution. Hence, the probability that A and A^* serve that request using active robots from two different children of the root is then $\frac{d_i-1}{d_i}$, where d_i denotes the r-degree.

A *trial* is the sequence of steps that A does starting with a single point at the root until it has a single point at the root the next time. A trial consists of at least four steps so the expected cost of A in stage i can now be expressed

as $E(A_i) \geq \sum_{j=1}^{d_i} E(A_i|j)$, where $E(A_i|j)$ denotes the conditional cost of A in stage i given that stage i consists of j trials. Then,

$$E(A_i|j) = k_{ij} \frac{1}{d_i - j + 1} \prod_{l=1}^{j-1} \frac{d_i - l}{d_i - l + 1} = \frac{k_{ij}}{d_i} \ , \tag{1}$$

where k_{ij} is the number of moves for A in stage i given that it consists of j trials. We know that $k_{ij} \geq j$ since each trial consists of at least four moves.

The corresponding expected cost of A^* in stage i given that A consists of j trials is

$$E(A_i^*|j) = \frac{k_{ij}/2 + 2}{d_i} \ , \tag{2}$$

since A_i^* will use only half the number of moves that A_i uses except in the last trial. In the last trial we can assume that both strategies use exactly four moves. In fact, we can let A_i know the moves of A_i^* when the last trial occurs so that A_i can follow A_i^*'s moves.

The ratio of the total expected costs for A and A^* is therefore bounded by

$$\begin{aligned}
\frac{E(A)}{E(A^*)} = \frac{\sum_{i=1}^{m} E(A_i)}{\sum_{i=1}^{m} E(A_i^*)} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{d_i} E(A_i|j)}{\sum_{i=1}^{m} \sum_{j=1}^{d_i} E(A_i^*|j)} = \\
\frac{\sum_{i=1}^{m} \sum_{j=1}^{d_i} \frac{k_{ij}}{d_i}}{\sum_{i=1}^{m} \sum_{j=1}^{d_i} \frac{\frac{k_{ij}}{2} + \frac{4}{2}}{d_i}} \geq \frac{\sum_{i=1}^{m} \frac{1}{d_i} \sum_{j=1}^{d_i} k_{ij}}{\sum_{i=1}^{m} \frac{1}{2d_i} \sum_{j=1}^{d_i} (k_{ij} + 4)} \geq \\
\frac{2 \cdot \sum_{i=1}^{m} \frac{1}{d_i} \sum_{j=1}^{d_i} j}{\sum_{i=1}^{m} \frac{1}{d_i} \sum_{j=1}^{d_i} (j+1)} \geq 2 \cdot \frac{\sum_{i=1}^{m} \frac{1}{d_i} \cdot \frac{d_i(d_i+1)}{2}}{\sum_{i=1}^{m} \frac{1}{d_i} \cdot \frac{d_i(d_i+3)}{2}} = \\
2 \cdot \frac{\sum_{i=1}^{m} (d_i + 1)}{\sum_{i=1}^{m} (d_i + 3)} \geq 2 \cdot \frac{\sum_{i=1}^{m} (i+1)}{\sum_{i=1}^{m} (i+3)} = \\
2 \cdot \frac{(m+3)}{(m+7)} \to 2 \ ,
\end{aligned} \tag{3}$$

as m increases, where m denote the number of stages. We have used the fact that $d_i \geq i$ since at each stage the r-degree increases by at least one.

Finally, note that there is one more case to handle, namely the case when m is constant. In this case, the ratio of the total expected costs for A and A^* depends on the cost of the last stage and is therefore bounded by

$$\begin{aligned}
\frac{E(A)}{E(A^*)} \geq \frac{\sum_{i=1}^{m} \frac{1}{d_i} \sum_{j=1}^{d_i} k_{ij}}{\sum_{i=1}^{m} \frac{1}{2d_i} \sum_{j=1}^{d_i} (k_{ij} + 4)} = \\
2 \cdot \frac{B + (\frac{1}{d_m}) \cdot \sum_{j=1}^{d_i} k_{mj}}{C + (\frac{1}{d_m}) \cdot \sum_{j=1}^{d_i} (k_{mj} + 4)} \to 2 \ ,
\end{aligned} \tag{4}$$

as n increases, where n denotes the number of requests, and B and C are constants. □

5 Time Dependent Freeze-tag

Let us change our problem specification for Freeze-tag. The robots are points in some general metric space and associated to each robot is a *release time*. The release time $t(r)$ specifies the first point in time when a robot r can be awakened and the awake robots become aware of r. An awake robot can move with at most unit speed to wake up sleeping robots. Hence, a request sequence is given by $\sigma = ((t(r_1), p(r_1)), \ldots, (t(r_n), p(r_n)))$, where $p(r_i)$ is the position of robot r_i. Initially one robot r_0 is awake and the objective is to find an awakening schedule that minimizes the time until all robots are awake, i.e., find the directed spanning tree of minimum height where the out degree of any point is at most two (except from the root point r_0 which has out degree one). We call such a tree an *optimal scheduling tree*. An optimal scheduling tree can be computed given a request sequence although the best known time complexity is exponential because of the NP-completeness of this problem [1].

The fact that the robots move in time requires us to use *continuous metric spaces* to accurately model the problem. This means that at any point in time a robot is positioned at some point in the metric space and robot motion is continuous.

We continue by providing a lower bound on the competitive ratio for *TDFT*.

Theorem 5. *There is a metric space such that no algorithm solves the time dependent Freeze-tag problem with competitive ratio* $7/3 - \epsilon$*, for any* $\epsilon > 0$*.*

Proof. Consider the graph structure of Fig. 3(a). It defines a continuous metric space where robots can move along the edges at unit speed. Points $p_1, \ldots, p_6$ are at distance three from the point p_0 having the initial active robot. The point p_7 is at distance six from p_0 and nine from the other ones.

The first request is $(0, p_0)$, i.e., at time zero a robot at p_0 is released thus giving us two active robots at p_0. These two robots are allowed to move along the edges to try to anticipate the release of subsequent robots. However, at time three the cruel adversary releases two robots, one on each of the points $p_1, \ldots, p_6$ that does not have an active robot on its corresponding edge. Without loss of generality we can assume that the requests are $(3, p_1)$ and $(3, p_2)$, thus giving us the situation depicted in Fig. 3(b)–(d).

The first active robot that serve a request will have to do so at the earliest at time six. Assume without loss of generality that this is p_1 and we look at the other robot r at time six. If r is further from p_2 than one, then nothing more happens giving a total time for the schedule of 7. The optimal schedule takes time three because the two robots can be at p_1 and p_2 respectively at time three.

On the other hand, if r is closer to p_2 than one, then a fourth request $(6, p_7)$ is generated by the cruel adversary. To serve this request, any of the robots will have to use at least a total of 14 time units. An optimal schedule will serve p_1

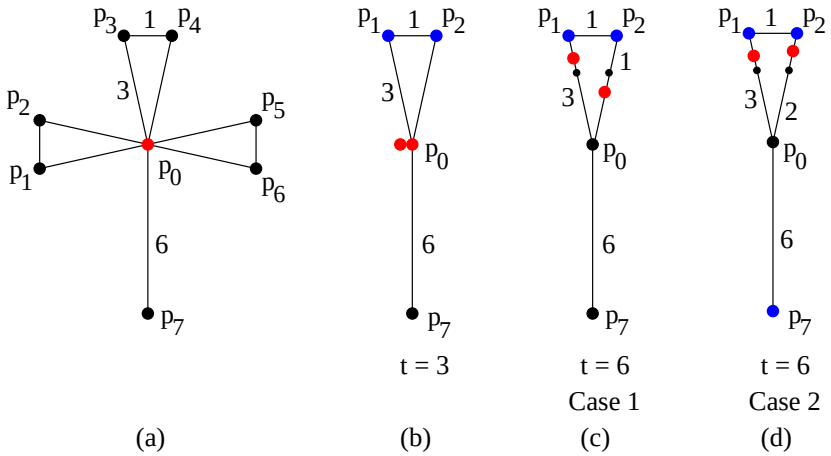

Fig. 3. Illustrating the proof of Theorem 5.

and p_2 in time four with one robot and the other will serve p_7 in six time units giving a total optimal cost of six. In both cases the ratio for any algorithm is at least 7/3. □

6 Conclusions

An interesting open problem is to investigate online algorithms for time dependent Freeze-tag. Does there exists a strategy with constant competitive ratio for this problem?

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 Symp. on Discrete Algorithms*, pages 568–577, 2002.
2. 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 Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 295–303, 2003.
3. A. Borodin and R. El-Yaniv. Online Computation and Competitive Analysis. Cambridge University Press, 1998.
4. E. Koutsoupias and C.H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.
5. M. Manasse, L.A. McGeoch, D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.
6. C.H. Papadimitriou, K. Steiglitz. Combinatorial Optimization, Algorithms and Complexity. Dover Publications, Inc., Mineola, New York, 1998.
7. M.O. Sztainberg, E.M. Arkin, M.A. Bender, and J.S.B. Mitchell. Analysis of Heuristics for the Freeze-Tag problem. In *Proc. 8th Scandinavian Workshop on Algorithm Theory*, pages 270–279, 2002.