An $O(1)$-Approximation Algorithm for the 2-Dimensional Geometric Freeze-Tag Problem

Ehsan Najafi Yazdi[a,*], Alireza Bagheri[a,b], Zahra Moezkarimi[a], Hamidreza Keshavarz[c]

[a] *Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran*
[b] *Computer Engineering Department, Tehran North Branch, Islamic Azad University, Tehran, Iran*
[c] *Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran*

Abstract

The problem of awaking a swarm of *asleep* robots, starting with only one *awake* robot, is named the *Freeze-Tag Problem (FTP)*. Waking up a robot is done by having an awake robot move to its position. Once a robot is awakened, it can assist in awaking others. In the FTP, the objective is to wake up all the robots in the shortest time possible. This problem is NP-Hard in general; the complexity of the geometric variant in the Euclidean plane is open. Arkin et al. [1] have given a constant-factor approximation algorithm, running in time $O(n \log n)$, for the *geometric FTP*, and utilized it as the basis of a PTAS. In this paper, we propose a simple $O(n)$-time constant-factor approximation algorithm, for the 2-dimensional geometric FTP.

Keywords: Freeze-Tag Problem, Broadcasting, Swarm robotics, Approximation algorithms

1. Introduction

The *Freeze-Tag Problem*, abbreviated as FTP, is an optimization problem arising in swarm robotics. The objective is to awaken a swarm of robots in the

*Corresponding author
Email addresses: ehsan@aut.ac.ir (Ehsan Najafi Yazdi), ar_bagheri@aut.ac.ir (Alireza Bagheri), zmoezkarimi@aut.ac.ir (Zahra Moezkarimi), Keshavarz.h@modares.ac.ir (Hamidreza Keshavarz)

shortest time possible. In the beginning, there is only one *awake* robot, and the rest are asleep. Waking up a robot is done by touching it by an awake robot, i.e., an awake robot should move to the asleep robot's position. Once a robot is awakened, it can assist in awakening others. The waking up process ends when the last robot is woken up [1]. The required time to wake up all the robots, is called the "*makespan*".

In general, an instance of the FTP can be seen as a weighted graph, in which the nodes represent robots, and distances between nodes are shown as the weights of the edges connecting them. In this way, the awakening schedule can be represented by a directed graph. The nodes represent robots, and each directed edge shows the movement-direction of a robot to wake up another. Since each robot is awakened exactly once, and an awake robot can only wake up one robot at a time, each solution is a binary spanning tree, called a "*wake-up tree*". The root of this tree represents the initially awake robot, and should have only one outgoing edge. The time required to wake up all the robots (makespan) is related to the depth of the tree, that is, the length of the longest path from the root to its leaves. In other words, the problem can be seen as finding a minimum-depth binary spanning tree, whose root is given and is forced to be degree-one. Arkin et al. [1] have proved that this is an NP-Hard problem.

An example of the FTP is represented in Figure 1-a, in which "A" is the first awake robot. An awaking schedule is shown by the arrows, which yields a makespan of 18. Figure 1-b represents the wake-up tree.

1.1. The Geometric Freeze-Tag Problem

One of the variants of the FTP is the geometric FTP, in which the robots are in a geometric space, and the distance between robots is the length of the line-segment connecting them. The complexity of the geometric FTP (e.g., in the Euclidean plane) is open. It is listed as Problem #35 on "The Open Problems Project" list [2]. However there are some approximation algorithms proposed to solve it.

In this paper, we consider the 2-dimensional geometric FTP, and propose a

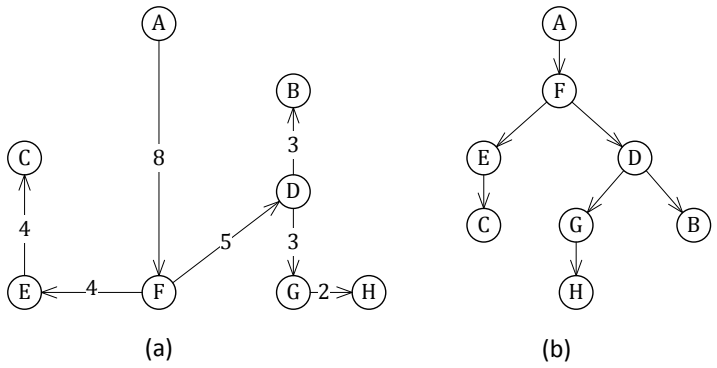

Figure 1: An example of the FTP (a), and its wake-up tree (b)

constant factor approximation algorithm for solving it. The proposed algorithm improves the previous results given by Arkin et al. [1].

1.2. Applications

Besides swarm robotics, one of the main applications of the FTP is in distribution and transfer of data, and arises when the sender and the receiver of the data should be physically near each other for transmittal. This proximity may be due to the security issues, or because of the cost of the bandwidth in wireless communications. It can be used for distributing any other duplicatable commodities, as well. It is closely related to the *Minimum maximum movement to perfect matchability (MatchMax) problem* that arises in the context of broadcasting or multicasting [3].

The FTP can also be considered as a hybrid of some problems from the areas of broadcasting, routing, scheduling, and network design [1]. For example, it is an instance of the *Cooperative Travelling Salesman problem (cTSP)* [4].

1.3. Related Work

As mentioned before, Arkin, Bender, Fekete, Mitchell, and Skutella [1] have proved that in general, the FTP is an NP-Hard problem. In another work, Arkin, Bender, Ge, He, and Mitchell [5] showed that even the FTP in unweighted graphs, is NP-hard. They gave an $O(1)-$approximation algorithm for

the FTP in unweighted graphs.

Sztainberg, Arkin, Bender, and Mitchell [6] have studied heuristics for the FTP. They showed that the greedy strategy gives a $\Theta((\log n)^{1-1/d})$ approximation bound, for the case that robots are in a D-dimensional space. More recently, Bucantanschi, Hoffmann, Hutson, and Kretchmar [7] proposed another heuristic called the "Neighborhood Search Technique" for the FTP.

Knemann, Levin, and Sinha [8] have studied the *degree-bounded minimum diameter spanning tree* problem, which can be seen as a generalization of the FTP. They have introduced an algorithm with an $O(\sqrt{\log n})$ approximation factor for the problem.

Moezkarimi and Bagheri [9] have considered a generalization of the FTP, called the *k-FTP*, in which there are k initially awake robots. They gave an approximation algorithm and a PTAS for the 2-FTP ($k = 2$).

Arkin et al. [1] have proposed a $(1+\epsilon)-$approximation algorithm (PTAS) for the geometric FTP, which runs in $O(2^{poly(1/\epsilon)} + n \log n)$ time. They also gave an $O(1)-$approximation algorithm, running in time $O(n \log n)$.

The latter algorithm is discussed in the next section. In section 3, we introduce an $O(1)-$approximation algorithm for the 2-dimensional geometric FTP, with linear running time. The concluding remarks are given in section 4.

2. Arkin et al.'s $O(1)-$Approximation Algorithm for the Geometric FTP

In this section, an approximation algorithm proposed by Arkin et al. [1] for the geometric FTP, is discussed. Its approximation factor is $O(1)$, and its time complexity is $O(n \log n)$. This algorithm generates a wake-up tree whose makespan is $O(radius(R))$, where R is the point-set of positions of robots, and $radius(R)$ is the radius of this set with respect to the first awake robot, i.e., the maximum distance of the first awake robot to any other member of R.

It is assumed that the robots are located in a $D-$dimensional space, and the speeds of all robots are constant and equal to one unit of speed, so that, a robot

requires l units of time to travel l units of distance.

2.1. Description of the Algorithm

For each point $v \in R$, the plane is partitioned into K sectors by rays emanating from v at angles of $0,\ 2\pi/K,\ 2(2\pi/K), ...,\ (K-1)(2\pi/K)$ radians. Let $u_j(v)$ denote the point (if any) of R in the jth sector that is closest to v. All the $u_j(v)$ points can be found for every $v \in R$, in $O(Kn \log n)$ time, with methods based on the standard Voronoi diagrams [10].

Then, for each $v \in R$, the points $u_j(v)$ are sorted by their distance from v. Assuming that for v, these points are $u_1,\ u_2, ...,\ u_K$ in ascending order, the awaking strategy is as follows: when the v robot wakes up, it travels the path $< v,\ u_1,\ u_2, ...,\ u_K >$, and awakes the nearest robot in each non-empty sector surrounding it (Figure 2). If a robot has been woken up before v reaches it, v skips awaking it.

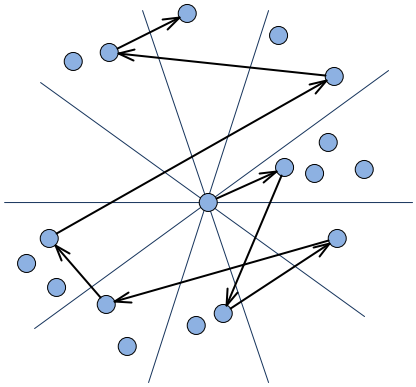

Figure 2: When v wakes up, it awakes each of the nearest robots in the K sectors surrounding it, in order of increasing distance from v.

2.2. The Performance of the Algorithm

Here, we discuss the performance of Arkin et al.'s algorithm. Assume that $G_K = (R, E_K)$ is a graph that connects each v point to the $u_j(v)$ points that are the nearest neighbors in the K sectors surrounding it. Hence, G_K is a "θ-graph" for $\theta = 2\pi/K$, and if $K \geq 9$, it is a "$t_\theta-$spanner" for $t_\theta = 1/(\cos\theta - \sin\theta)$ [11].

This means that the graph distance of any two nodes of the graph G_K for $K \geq 9$, is not greater than $1/(\cos\frac{2\pi}{K} - \sin\frac{2\pi}{K})$ times of the Euclidean distance of those two nodes.

Assume the first robot is at the point v_0, and the last robot that is woken up by this algorithm, is at the point v_l. How much time is required to wake up v_l? We know that if the robot v is woken up at time t, any u_j neighbors of v in the graph G_K, is reached in a time shorter than or equal to $t + \xi$, where ξ is the length of the path $< v,\ u_1,\ u_2, ...,\ u_K >$. Denote Euclidean distance and graph distance of u and v by $d(u, v)$ and $d_{G_K}(u, v)$ respectively. With a simple induction and the triangle inequality, we have:

$$\xi \leq (2j - 1) \cdot d(v, u_j) \leq (2K - 1) \cdot d(v, u_j)$$

Thus, v_l is reached in a time that is no longer than $(2K - 1)\ d_{G_K}(v_0, v_l)$, and because the graph is a "$t_\theta-$spanner" graph for $K \geq 9$:

$$\begin{aligned} makespan(R) &\leq (2K - 1)\ d_{G_K}(v_0, v_l) \\ &\leq \frac{2K - 1}{\cos(2\pi/K) - \sin(2\pi/K)}\ d(v_0, v_l) \end{aligned}$$

So, v_l is woken up within the time of $O(d(v_0, v_l))$, and since $d(v_0, v_l)$ is a lower bound for the optimum makespan, this algorithm is an $O(1)-$approximation. The time complexity of the algorithm is $O(n \log n)$ for a constant K. We refer the reader to [1] for more details.

As mentioned above, Arkin et al.'s algorithm guarantees a constant approximation factor. The upper bound of this factor, is the function $\frac{2K - 1}{\cos(2\pi/K) - \sin(2\pi/K)}$ for $K \geq 9$. As depicted in Figure 3, which graphs the function, this bound is greater than 57, i.e., the algorithm does not guarantee any approximation factors lower than 57.

3. The Proposed Algorithm for the Geometric FTP

In this section, we give a constant-factor, linear-time approximation algorithm for the 2-dimensional geometric FTP. The main idea of our algorithm is

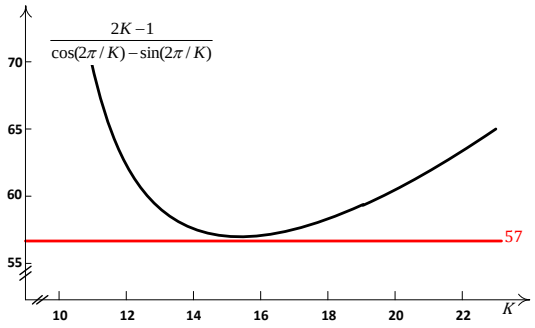

Figure 3: The function $\frac{2K-1}{\cos(2\pi/K)-\sin(2\pi/K)}$

to split the area, in which the robots are, and to solve the problem in a divide-and-conquer manner, by matching asleep robots to awake ones.

Assume there is one awake robot named s, and R is the set of n asleep robots (note that in this section, R does not include s); $d = diam(R)$ is the diameter of R, that is, the longest Euclidean distance between any two asleep robots; and d_s is the radius of R with respect to s, that is, the maximum Euclidean distance from s to any member of R.

If so, an $a \times b$−rectangle with $a, b \leq d$, can be found, which encompasses positions of all members of R, while its sides are parallel to the axes. Let A be such a rectangle (Figure 4).

Our algorithm, *ApproxFTP*, inputs s as the only awake robot, and R as the set of the asleep robots, and provides an awaking schedule.

If $|R| \leq 3$, s wakes up an (arbitrary) asleep robot. This is done in a time not longer than d_s. Then, there will be at most two asleep robots remaining, which can be woken up by the two awake robots working in parallel, in a time not more than $\sqrt{2}\ d$. (Since the robots are in a rectangle whose length and width are at most d.) Therefore, the whole awakening process can be done within time $d_s + \sqrt{2}\ d$.

For $|R| \geq 4$, the algorithm works recursively. First, we find the sets R' and R'', by dividing A and R as described in the following: (See Algorithm 1 for the pseudocode.)

Let $m(x_m, y_m) \in R$ be the (left) median of R in the lexicographical order based on x and then y coordinates. (Which means, if the members of R are sorted in order of their x values, and members with equal x value, are sorted in order of their y values, m will be the $\lceil \frac{n}{2} \rceil$th member.) A vertical line that passes through m, splits A into two sub-rectangles. At least, one of these rectangles, say A', has a horizontal side of length $a' \leq d/2$ (Figure 4).

Denote by R', the set of the robots of R that are located in A', including the robot m, if n is even and A' is the left sub-rectangle, and excluding m otherwise. It is obvious that R' has $\lfloor \frac{n}{2} \rfloor$ members.

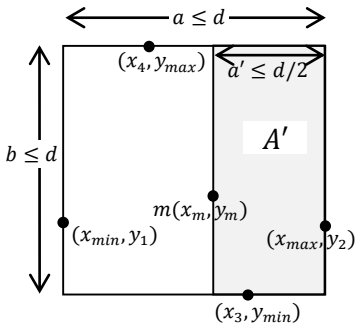

Figure 4: Splitting the rectangle A vertically.

Now, let $m'(x_{m'}, y_{m'}) \in R'$ be the (left) median of R' in the lexicographical order based on y and then x coordinates. Like the previous phase, a horizontal line that passes through m' divides A' into two rectangles, and at least one of them, say A'', has a vertical side of length $b'' \leq d/2$. Note that the horizontal side of A'' is also of length $a'' \leq d/2$ (Figure 5).

Denote by R'', the set of the robots of R that are located in A'', including the robot m' if $\lfloor \frac{n}{2} \rfloor$ is even and A'' is the bottom sub-rectangle, and excluding m' otherwise. Obviously R'' has $\lfloor \frac{|R'|}{2} \rfloor \leq \frac{n}{4}$ members.

Then, the algorithm runs with s and R'' as inputs, to awaken all the robots of R''. Notice that the awake robots are not necessarily in their initial places, but in (or very close to) places where some robots of R'' were initially there. Thus, all of the awake robots, including s, are located in A''.

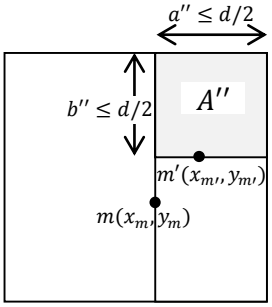

Figure 5: Splitting the rectangle A' horizontally.

Now, the number of the asleep robots remaining in R', is less than or equal to the number of the awake robots, which are the members of $R'' \cup \{s\}$:

$$|R' \setminus R''| = |R'| - \lfloor \frac{|R'|}{2} \rfloor \leq |R''| + 1$$

So, we assign to any robot of $R' \setminus R''$, one of the robots of $R'' \cup \{s\}$. To awake all the robots of R', each of the asleep robots of $R' \setminus R''$ can be awakened by its assigned robot of $R'' \cup \{s\}$. Since the robots work in parallel, the time needed for this phase is bounded by the diagonal of A', which is at most $\sqrt{5}/2\ d$.

After this phase, s and all the robots of R' are awake, and all of them are located in A'. Here again, since the number of robots of $R \setminus R'$ is not more than the number of the awake robots $R' \cup \{s\}$, they can be awakened in parallel in one phase. And the time needed to do this, is bounded by the diagonal of A, which is at most $\sqrt{2}\ d$.

3.1. The Approximation Factor of ApproxFTP

In this section, we assess the approximation factor of our proposed algorithm.

Lemma 1. *Suppose that n asleep robots are located inside a rectangle, whose length and width are shorter than or equal to d, and there is an awake robot whose distance from the asleep robots is at most d_s. By applying ApproxFTP, all asleep robots can be awaken within time $(2\sqrt{2} + \sqrt{5})d + d_s$.*

Proof. If $makespan(n, d)$ is the required time to wake up n robots with the

Algorithm 1 ApproxFTP(R, s)

Input. R: Set of asleep robots, s: An awake robot

if $|R| \leq 3$ **then**

Assign s to awake an (arbitrary) member r of R.

Assign $\{s, r\}$ to awake the members of $R \setminus \{s\}$.

return

else ▷ $|R| \geq 4$

$m(x_m, y_m) \leftarrow \lfloor \frac{|R|}{2} \rfloor$th member of R, in the lexicographical order based on (x, y) coordinates

if $(x_m - min\{x_r | r \in R\}) \leq (max\{x_r | r \in R\} - x_m)$ **then**

$R' \leftarrow \{\ r \in R \mid x_r < x_m\ or\ (x_r = x_m\ and\ y_r < y_m)\ \}$

if $|R|$ is even **then** $R' \leftarrow R' \cup \{m\}$ **end if**

else

$R' \leftarrow \{\ r \in R \mid x_r > x_m\ or\ (x_r = x_m\ and\ y_r > y_m)\ \}$

end if

$m'(x_{m'}, y_{m'}) \leftarrow \lfloor \frac{|R'|}{2} \rfloor$th member of R', in the lexicographical order based on (y, x) coordinates

if $(y_{m'} - min\{y_r | r \in R'\}) \leq (max\{y_r | r \in R'\} - y_{m'})$ **then**

$R'' \leftarrow \{\ r \in R' \mid y_r < y_{m'}\ or\ (y_r = y_{m'}\ and\ x_r < x_{m'})\ \}$

if $|R'|$ is even **then** $R'' \leftarrow R'' \cup \{m'\}$ **end if**

else

$R'' \leftarrow \{\ r \in R' \mid y_r > y_{m'}\ or\ (y_r = y_{m'}\ and\ x_r > x_{m'})\ \}$

end if

ApproxFTP(R'',s)

Assign the members of $R'' \cup \{s\}$ to awake the members of $R' \setminus R''$.

Assign the members of $R' \cup \{s\}$ to awake the members of $R \setminus R'$.

end if

return

aforementioned conditions, for $n \geq 4$ we have:

$$\begin{aligned} makespan(n,d) &\leq makespan(\left\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rfloor, \frac{d}{2}) + \sqrt{2}d + \frac{\sqrt{5}}{2}d \\ &\leq makespan(\frac{n}{4}, \frac{d}{2}) + (\sqrt{2} + \frac{\sqrt{5}}{2})d \end{aligned}$$

and for $n \leq 3$:

$$makespan(n,d) \leq \sqrt{2}\ d + d_s$$

By solving this recurrence relation we have:

$$\begin{aligned} makespan(n,d) &\leq (2\sqrt{2} + \sqrt{5})d - (\sqrt{2} + \frac{\sqrt{5}}{2})2^{-\lfloor \log_4 n \rfloor} + d_s \\ &\leq (2\sqrt{2} + \sqrt{5})d + d_s \end{aligned}$$

□

It is worth mentioning, that in the first call of the algorithm, in which $d = diam(R)$, the distance between each pair of robots is less than or equal to d, while in the next calls, the distance of two robots can be as long as the diagonal of the encompassing rectangle. This is the same distance that has appeared as $\sqrt{5}/2\ d$ and $\sqrt{2}\ d$ in our computations. Only for the first call, we can be assured that the distance of the robots is at most d, instead of $\sqrt{5}/2\ d$ and $\sqrt{2}d$. The following lemma uses this.

Lemma 2. *By applying ApproxFTP algorithm, the required time for awaking n asleep robots, whose positions are given by a set R, using one awake robot, whose distance from the asleep robots is not greater than d_s, is at most $(2 + \sqrt{2} + \sqrt{5}/2)diam(R) + d_s$.*

Proof. In the first call of the algorithm, the distance of the robots from each other is at most $diam(R)$. So, after awaking all the robots of R'', the other robots of R can be woken up in two phases, each of a time not more than $diam(R)$. Hence:

$$makespan(R) \leq makespan(\frac{n}{4}, \frac{diam(R)}{2}) + 2diam(R)$$

For next call of the algorithm, we use Lemma 1:

$$makespan(\frac{n}{4}, \frac{diam(R)}{2}) \leq (2\sqrt{2}+\sqrt{5})\frac{diam(R)}{2} + d_s$$

and as a result:

$$\begin{aligned} makespan(R) &\leq (2\sqrt{2}+\sqrt{5})\frac{diam(R)}{2} + d_s + 2\ diam(R) \\ &\leq (2+\sqrt{2}+\frac{\sqrt{5}}{2})diam(R) + d_s \end{aligned}$$

□

Theorem 3. *ApproxFTP is an approximation algorithm, with a constant approximation factor less than* 10.1.

Proof. In the previous lemma, it was proven that:

$$makespan(R) \leq (2+\sqrt{2}+\frac{\sqrt{5}}{2})diam(R) + d_s$$

We know that $diam(R) \leq 2d_s$, because, if we draw a circle with s as the center, and d_s as the radius, it will encompass all of the robots of R. Hence:

$$makespan(R) \leq (5+2\sqrt{2}+\sqrt{5})d_s < 10.1\ d_s$$

Since d_s is a lower bound for the optimum makespan, the approximation factor of *ApproxFTP* that is less than 10.1. □

3.2. The Time Complexity of ApproxFTP

To compute the time complexity of this algorithm, it should be noted, that finding the median of a set with n members, can be done in $O(n)$ time [12]. So, the algorithm can find A', A'', R' and R'' in $O(n)$ time. Then, the algorithm is called with R'' as its input. The number of the members of R'' is less than or equal to $n/4$. After that, the algorithm finds matchings between the members of $R' \setminus R''$ and $R'' \cup \{s\}$, and the members of $R \setminus R'$ and $R' \cup \{s\}$. Theses phases can also be done in $O(n)$.

So, if the running time of algorithm for $|R| = n$ is shown as $T(n)$,

$$T(n) = T(\frac{n}{4}) + O(n) \qquad n \geq 4$$

$$T(n) = O(1) \qquad n \leq 3$$

By solving this recursive relation, we find that $T(n) = O(n)$.

4. Conclusion

In this paper we considered the Freeze-Tag Problem in the Euclidean plane, whose time complexity is open. We gave a simple linear-time algorithm that achieves an approximation factor less than 10.1. Our algorithm can be generalized to any fixed dimension with the same time complexity, but the approximation factor would depend on dimension.

References

[1] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. B. Mitchell, M. Skutella, The freeze-tag problem: How to wake up a swarm of robots, Algorithmica 46 (2) (2006) 193–221. doi:10.1007/s00453-006-1206-1.

[2] E. D. Demaine, J. S. B. Mitchell, J. O'Rourke, The open problems project. URL http://maven.smith.edu/~orourke/TOPP/

[3] E. D. Demaine, M. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, M. Zadimoghaddam, Minimizing movement, ACM Transactions on Algorithms 5 (3) (2009) 1–30. doi:10.1145/1541885.1541891.

[4] A. Armona, A. Avidor, O. Schwartz, Cooperative tsp, Theoretical Computer Science 411 (31-33) (2010) 2847–2863. doi:10.1016/j.tcs.2010.04.016.

[5] E. M. Arkin, M. A. Bender, D. Ge, S. He, J. S. B. Mitchell, Improved approximation algorithms for the freeze-tag problem, in: Proc. 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, NY, USA, 2003, pp. 295–303. doi:10.1145/777412.777465.

[6] M. O. Sztainberg, E. M. Arkin, M. A. Bender, J. S. B. Mitchell, Theoretical and experimental analysis of heuristics for the "freeze-tag" robot awakening problem, IEEE Transactions on Robotics 20 (4) (2004) 691–701. doi:10.1109/TRO.2004.829439.

[7] D. Bucantanschi, B. Hoffmann, K. R. Hutson, R. M. Kretchmar, A Neighborhood Search Technique for the Freeze Tag Problem, Vol. 37 of Operations Research/Computer Science Interfaces, Springer, New York, NY, USA, 2007, pp. 97–113. doi:10.1007/978-0-387-48793-9_7.

[8] J. Könemann, A. Levin, A. Sinha, Approximating the degree-bounded minimum diameter spanning tree problem, Algorithmica 41 (2) (2004) 117–129. doi:10.1007/s00453-004-1121-2.

[9] Z. Moezkarimi, A. Bagheri, A ptas for geometric 2-ftp, Information Processing Letters 114 (12) (2014) 670–675. doi:10.1016/j.ipl.2014.06.017.

[10] K. L. Clarkson, Approximation algorithms for shortest path motion planning, in: Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, NY, USA, 1987, pp. 56–65. doi:10.1145/28395.28402.

[11] J. M. Keil, C. A. Gutwin, Classes of graphs which approximate the complete euclidean graph, Discrete & Computational Geometry 7 (1) (1992) 13–28. doi:10.1007/BF02187821.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, McGraw-Hill Higher Education, 2009.