ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 5, No.17 , September 2015
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Linear time constant factor approximation algorithm for the Euclidean "Freeze-Tag" robot awakening problem

Mohammad Javad Namazifar[1], Alireza Bagheri[2] and Keivan Borna[3]

[1] Computer Engineering & IT Department, Qazvin Branch, Islamic Azad University
Qazvin, Iran
MJ.Namazifar@qiau.ac.ir

[2] Computer Engineering & IT Department, Amirkabir University of Technology
Tehran, Iran
ar.bagheri@aut.ac.ir

[3] Department of Computer Science, Kharazmi University, Faculty of Mathematics and Computer Science
Tehran, Iran
borna@khu.ac.ir

Abstract

The Freeze-Tag Problem (FTP) arises in the study of swarm robotics. The FTP is a combinatorial optimization problem that starts by locating a set of robots in a Euclidean plane. Here, we are given a swarm of n asleep (frozen or inactive) robots and a single awake (active) robot. In order to activate an inactive robot in FTP, the active robot should either be in the physical proximity to the inactive robot or "touch" it. The new activated robot starts moving and can wake up other inactive robots. The goal is to find an optimal activating schedule with the minimum time required for activating all robots. In general, FTP is an NP-Hard problem and in the Euclidean space is an open problem. In this paper, we present a recursive approximation algorithm with a constant approximation factor and a linear running time for the Euclidean Freeze-Tag Problem

Keywords*: Freeze Tag Problem, Recursive Algorithm, Optimization, Swarm Robotics, Computational Geometry.*

1. Introduction

FTP was first introduced by Joseph Mitchell at the Fall Workshop on Computational Geometry. FTP is a combinatorial optimization problem that arises in the field of swarm robotics. At the start, in this swarm, there is only one "awake" or active robot (v_r), and all the other robots are "asleep". The goal is to awaken all of the inactive robots by the first robot in the shortest possible time. Activating the robots occurs only by touching or being in the physical proximity to the asleep robots. Once a robot is awakened, it can assist in activating other sleeping robots. The problem will only be ended when all the robots are awakened [1].

FTP, has its name from a children's game called "*Freeze Tag*", where a player is chosen to be "it" who tries to tag (touch) other players. Once players are tagged by "it" they must "*freeze*" in place until an uncaptured player tag (un-freeze) them and thereby return them to the game. The traditional FTP takes the special case where there is only one uncaptured player remaining and 'it' is incapacitated and will not be attempting to re-freeze players. The one remaining uncaptured player must release all the frozen players in the shortest possible time (in scheduling terminology, this is called the makespan). Once a player is unfrozen, he or she assists by unfreezing other frozen players. As in Arkin et al's [2] research on FTP the frozen players are replaced by stationary robots and 'it' is removed from the problem. This revised terminology is used in this research.

The problem can be demonstrated by a directed graph in which robots sleep at the vertices and the edges show the direction of an awake robot's movement in order to awaken an asleep robot. Since, each robot can be awakened just once, and each awake robot can wake up only one asleep robot at a time, the solution to the problem would be a directed binary tree, with v_r as its root [2], which is obtained as follows:

In order to awaken an asleep robot from a set of asleep robots R, robot v_r moves to the location of the asleep robot r_i. Once the robot v_r reaches to the proximity of the robot r_i, robot r_i is awakened then both robots v_r and r_i start to move from the location of robot r_i towards the other asleep robots. The process of awakening continues until the last robot is awakened.

According to the structure of this tree, obtaining a solution to the FTP requires finding a minimum spanning tree in a complete weighted graph, provided that the out- degree of the root node must be equal to 1. The resulted tree determines the schedule and order of the process of awakening the robots, and is therefore called an "awakening tree" [2]. The objective of the FTP is to find an awakening tree that makes the "*makespan*" (i.e. the

length of the path between the root node and the furthest leaf) as short as possible [3].

Arkin et al.'s study proved that FTP is an NP-Hard problem, even for the case of star graphs with an equal number of robots at each vertex [2]. Many solutions for FTP have been proposed including approximation algorithms, heuristic strategies, local search and computational intelligence.

Applications of the FTP arise in the context of distributing and transmitting data (or any other copiable commodity), where transmitting data requires robots to be in physical proximity to each other. Each robot assists in transmitting information, after copying them. The physical proximity approach in transmitting data is used when the wireless communication has high bandwidth cost or security risk. In fact, FTP is a combination of issues in the areas of distributing data, routing, scheduling, and network design.

While the FTP bears some resemblance to some problems like the minimum broadcast time [4], the minimum multicast time [5], and the minimum gossip time [6] , it has also some significant differences with them. For example, the time complexity of a data distribution problem in tree networks has a polynomial order while for a pretty simple instance of a weighted star graph the FTP is NP-hard [2].

In [7] an approximation algorithm with an approximation factor of 10.1 is presented. This paper suggests a recursive approximation algorithm for FTP, which is able to improve the approximation factor presented in [7] and its time complexity is linear.

The remainder of this paper is structured as follows. Section 2 looks at some related works about the problem. Section 3, at first describes the modeling of the problem, then clarifies the definitions needed to solve the problem, and finally discusses the idea of "divide and conquer" which is used to solve the FTP problem. Next, Section 4 explains the time complexity of the algorithm. Finally, Section 5 concludes the study with a comparison between the results achieved from the suggested algorithm with those of other available algorithms.

2. Related Work

Existing solutions to FTP can be classified as approximation algorithms, heuristic strategies, local search and computational intelligence.

Some solutions are discussed based on star graphs and ultra-metric in [2] and online FTP is studied in [8]. The heuristic strategies for this problem are divided into two groups: the greedy strategies and the sector-based strategies. The main idea in greedy strategies is that an awake robot selects and awakens the closest asleep robot in each step. The greedy strategy is not a fully defined heuristic. The reason behind choosing the closest robot is to achieve parallelism in the process of awakening the robots. The weakness of greedy strategy is that sometimes a robot prefers to travel a longer distance to obtain a better payoff. Best schedules are not always obtainable by selecting the closest robot.

The main idea in greedy strategy is to select the nearest robot in each step (an awaken robot selects the nearest robot for awakening). Therefore, the purpose behind choosing the nearest robot is to create a parallelism in the process of awakening the robots. But, the weakness of greedy strategy is that sometimes a robot prefers to travel a longer distance to obtain a better payoff. Best schedules are not always obtainable by selecting the closest robot.

Thus, some other heuristic strategies were designed, which are introduced by the term *sector-based strategies* in [9].We call these strategies sector-based strategies because the authors use the sectors around the robots for making decisions. The robot can detect the closest robot within each of its sectors. Three methods that have been proposed in [9] are Bang-for-the-Buck, Random Sector Selection and Opposite Cone.

Local and computational intelligence solutions try to optimize the awakening trees of greedy strategies. Local search is used by replacement paths method [3] to improve the obtained answers for FTP.

In [10] a simulated annealing algorithm is introduced, which receives the awakening tree, obtained through the greedy fixed method, as its input and optimizes its awakening tree.

In addition, [11] employs a genetic method to solve FTP problem, and [12] introduces a novel approach to FTP by DNA computations, in which thermodynamic properties of DNA and other biochemical reactions are used to obtain optimum schedule for FTP.

Furthermore, [13] deals with an extended type of FTP, called k-FTP. At the beginning of the problem, there are k active robots in k-FTP. In [13], first it was denoted that the approximation algorithm with a constant factor and a running time of $O(n \log n)$, introduced by Arkin et al. in [2], is also expandable for k-FTP. Then a PTAS was performed for 2-FTP, i.e. when k=2.

Arkin et al. [2] proposed an approximation algorithm with $O(1)$ as its approximation factor and $O(n \log n)$ as its running time for FTP, in a Euclidean plane. This algorithm creates an awakening tree with a makespan of $O(diam(R))$, where R stands for the number of nodes (robots) and $diam(R)$ is the radius of total nodes R, that is the maximum distance between the first active robot and the other robots. It is assumed that the robots have k peripheral sensors, which are separated by lines at $\frac{(K-1)2\pi}{k}, \dots, \frac{4\pi}{k}, \frac{2\pi}{k}, 0$ angles.

Each robot is capable of measuring its distance to the nearest robot at each sector. So that, each robot can have up to k nearest robots at any moment. If v is a robot in the

set of robots R then, $u_j(v)$ denotes the nearest point (if any) of R in the jth sector $(1 \le j \le k)$. If there are no points of R in the jth sector of v, then $u_j(v)$ is undefined. All $u_j(v)$ points for all j and all $v \in R$, in the total time of $O(kn\,log\,n)$are computed by the help of standard Voronoi diagram-based methods [14].

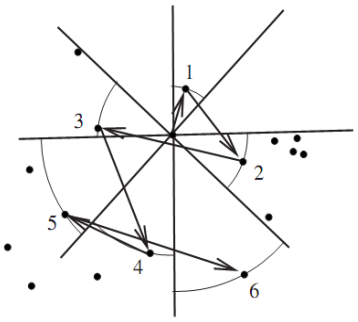


Fig.1. when robot v is awakened, it starts to awake the nearest robot at each k sector in order of the distance [2]

$u_j(v)$ Points for each $v \in R$ are sorted based on their distance from v; let these points be $u_1, u_2, \dots, u_k$ in sorted order by distance from v (Figure 1).

Awakening strategy implies that when robot v is awakened, it starts to awake other nearest robots in non-empty sectors by following $v, u_1, u_2, \dots, u_k$ path. Let $G_k = (R, E_k)$ be graph that links each point v to the points $u_j(v)$ that are its nearest neighbors in k sectors about v.

Such a graph G_k is known as a $\theta-graph$, for $(\theta = \frac{2\pi}{K})$ and is known to be a $t_\theta - spanner$, for values of $K \ge 9$ $(\mathrm{t}_\theta = \frac{1}{\cos\theta - \sin\theta})$ [2].

That is, the length of nearest path between two nodes of graph G_k when $K \ge 9$ is not more than $\frac{1}{cos(2\pi/k) - sin(2\pi/k)}$ times of Euclidean distance between two nodes. Assume that the first awaken robot be at point v_0 and the last one unfrozen robot is at point v_l, what is the running time of v_l? We know that if some point v is reached at t, then any neighbor (u_j) of v in graph G_k will be reached by $distance \le t + \xi$, where ξ is the length of the path $v, u_1, u_2, \dots, u_k$. With $d(u, v)$ as Euclidean distance between the two graphs, a simple inductive and triangle inequality yield:

$$\xi \le (2j - 1).d(v, u_j) \le (2K - 1).d(v, u_j)$$

Thus, the v_0 will reach v_l within a distance of at most $(2K - 1)$ times the distance from v_0 to v_l in G_k $(d_{GK}(v_0, v_l))$ considering G_k is $t_\theta - spanner$ and for constant $k \ge 9$, distances are less than or equal $\frac{1}{\cos(2\pi/k) - \sin(2\pi/k)}$ for Euclidean distance v_0 and v_l. That is

$$makespan \le 2(k-1)d_{GK}(v_0, v_l)$$
$$\le \frac{1}{cos\,(2\pi/k) - sin\,(2\pi/k)}\, d(v_0, v_l)$$

Thus, v_l is awakened at $O(d(v_0, v_l))$, and because $d(v_0, v_l)$ is a lower bound on the optimal makespan, the above algorithm is an $O(1) - \mathrm{Approximation}$. The time complexity of the algorithm for each fixed k is $\mathrm{O}(kn\,log\,n)$.

3. A recursive approximation algorithm for awakening robots in Euclidean space

In this paper, a recursive algorithm for FTP problem is proposed. The main idea in this algorithm is to divide a set of asleep robots into smaller subsets and once awaken the smallest set of robots by the first active robot. Afterward the remaining asleep robots at each step are awakened simultaneously by the awakened robots in the previous step. This process continues until all the robots are awakened. Time complexity of the algorithm is linear.

3.1 Preliminaries (Modeling the Problem)

Let R be a set of n asleep robots in a Euclidean space. Robots are distributed at a two-dimensional plane and the distance between them is determined based on the Euclidean distance. Each robot in R corresponds to a point in the Euclidean space. In this study the offline version of the problem is considered in which each active robot knows the position of all the other robots and can coordinate its moves with the other robots. The robots move at a constant velocity (v = 1 unit) - that is to say, the time needed to pass the distance x by each robot is x time unit.

At the beginning of the problem, there is a set of asleep robots and a source robot v_r which is active. Let R be the total points corresponding to asleep robot and $d = Diam(R)$ be the diameter of the set of points R. In other words, consider d the Euclidean distance between two farthest asleep robots and d_v as the maximum Euclidean distance between the source robot v_r and the farthest asleep robot in R.

3.2 Determining the Space Containing Robots

The space Containing Robots, in fact a bounding box enclosing set points R based on 2D. In each step of recursive algorithm, first of all finding the rectangle of minimum area in which all remaining asleep robot can be contained. The bounding box used is defined by the

extreme coordinate values of the points with respect to global coordinate system. So the farthest points in the set of points R must be found. If the four extreme coordinate values of the points in the set of R, that is (x_{min}, y_1) and (x_{max}, y_2) represent minimum and maximum values of x respectively, and (x_3, y_{min}) and (x_4, y_{max}) represent minimum and maximum value of y respectively. The rectangle $\big((x_{min}, y_{min}) , (x_{\max}, y_{max}) \big)$ we call the Space Containing Robots, such that set of points R can be placed in an imaginary rectangle $a \times b$ $(a, b \leq d)$, that has sides parallel to the coordinate axes.

3.3 Dividing the Space Containing Robots

In the *divide-and-conquer* strategy, trying to divide the set asleep robots into the smaller subsets and then awaken the smallest set of robots by the source robot. In other words, in each step of recursive algorithm, the set R is divided into two subset and the subset with more density in the imaginary rectangle is selected for the next step. The division continues until the number of robots in a set is more than four.

Considering that at each step of calling, the set of point R is divided into two subsets, the division process leads to a binary tree with depth of "depth". At the first step of calling, the depth of the binary tree is equal to 1. We call this binary tree, "*space division tree*".

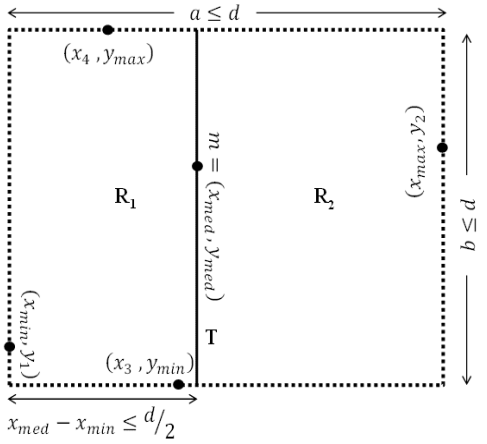
The space division can be performed *vertically* or *horizontally* depending on the *depth*, the depth of the division tree in each step, being odd or even. In the vertical division, the midpoint of R is obtained in terms of x; it means that, if R is arranged based on x, then the $\left[\frac{n}{2}\right] th$ point is called m (if the lowest $x - coordinate$ exists in more than one point in the set, the point with the lowest $y - coordinate$ should be chosen out of the other candidates.).

Figure 2, shows a schema of dividing the set R, in section a, the vertical imaginary line T, which crosses m, divides the rectangle into two parts, which the length of at least one of them is equal or less than $\frac{d}{2}$. The set of the points located in the smaller rectangle is denoted by R_1, and m along with the rest of the points are located in the set R_2. Afterward, the algorithm is called, with R_1, maximum distance between two asleep robots in the set, and $depth + 1$ as the inputs. As it is obvious, since the midpoint m is added to set R_2 at the end of this stage, the set R_1 has $\left[\frac{n}{2}\right] - 1$ points.

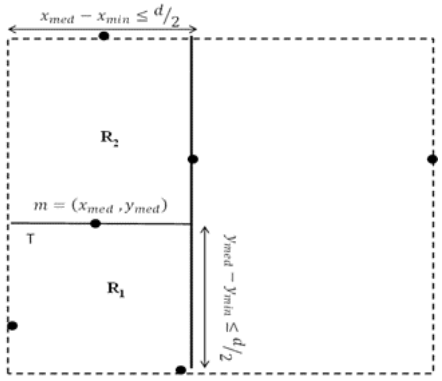
At the next stage, dividing with respect to value of *depth* performed horizontally. So, the midpoint m is obtained for R based on y (if the lowest $y - coordinate$ exists in more than one point in the set, the point with the lowest $x - coordinate$ out of the candidates should be chosen.)

In section b of Figure 2, This time the horizontal imaginary line T that crosses m divides the rectangle to two vertical sections so that the width of at least one of them is equal or less than $\frac{d}{2}$, which is called R_1 and m and the rest of the points are considered as R_2.

The set R_1 has maximum $n/4$ asleep robots, at the end of this stage. Then, the algorithm is called, with R_1, $\frac{d}{2}$ and $depth + 1$ as the inputs.



a. dividing vertically



b. dividing horizontally

Fig.2. a schematic view of division of the robots space into R_1 and R_2 sets by the line T crossing from point m; set R_1 is selected.

3.4 Recursive Awakening Algorithm

The algorithm receives the source robot v_r, the set of asleep robots R, diameter of the set R (d), and the depth of the division tree "*depth*", which is equal to 1 at the first call, as its inputs.

Then, the recursive algorithm is performed as follows based on members of the set R and depth of division tree at each stage (e.g. stage i where $1 \le i \le depth$).
If at the depth i, $|R| < 4$ then, at first, robot v_r awakens one of the asleep robots, which is done at most by the time d_v. Finally, At most two robots remain asleep, which can be awakened by two awaken robots at the same time.
The awakening time of the two remaining robots will be as follows:

- When depth i is odd, it will be done at the maximum time of $\frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}} d$ (since the robots are placed at a square with maximum length of d and the depth of i, the distance between no two robots is more than $\frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}} d$). Hence, when i is odd, the maximum time of awakening will be $d_v + \frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}} d$.
- When depth i is even, it will be done at the maximum time of $\frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}} d$ (since the robots are placed at a rectangle with maximum length of d and maximum width of $\frac{d}{2}$ and note at the depth of i, the distance between no two robots is more than $\frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}} d$). Hence, when i is even, the maximum time of awakening will be $d_v + \frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}} d$.

As a result, the awakening time at depth i, by the source robot v_r, when the number of asleep robots are less than four ($|R| < 4$) is:

$$makespan(R, d, i) \le \begin{cases} \frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}} d + d_v & if\ i\ is\ odd \\ \frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}} d + d_v & if\ i\ is\ even \end{cases} \quad |R| < 4 \qquad (1$$

$$1 \le i \le depth$$

If the number of asleep robots at the depth of i is more than or equal to four ($|R| \ge 4$), the algorithm works recursively. First, as mentioned above, a rectangle, encompassing all the asleep robots, is assumed for the set R. Afterward, when the "*depth*" is odd a midpoint is obtained based on x and the vertical line T crossing the midpoint is drawn. For even "*depth*" the midpoint is determined based on y and the horizontal line T crossing the midpoint is drawn. Therefore, the set R is divided into two sub-sets of R_1 and R_2 . Then the algorithm is performed with the inputs from R_1, diameter of R_1 and the depth of $i + 1$. This process continues until the number of remaining asleep robots is less than four. On the other hand:

if depth is odd then
  if $\left((x_{med} - x_{min}) \le d / 2\right)$ *then*
    $R_1 = \{(x, y) \in R |\ (x < x_{med}) or\ (x = x_{med}\ and\ y < y_{med})\}$
  else
    $R_1 = \{(x, y) \in R |\ (x > x_{med}) or\ (x = x_{med}\ and\ y > y_{med})\}$
else
  if $\left((y_{med} - y_{min}) \le d / 2\right)$ *then*
    $R_1 = \{(x, y) \in R |\ (y < y_{med})\ or\ (y = y_{med}\ and\ x < x_{med})\}$
  else
    $R_1 = \{(x, y) \in R |\ (y > y_{med})\ or\ (y = y_{med}\ and\ x > x_{med})\}$
$R_2 = R - R_1$

Given that there are $R_1 \cup \{v_r\}$ awake robots at depth i, the number of remaining asleep robots in R at each depth is less than or equal to the number of awake robots at the same depth. Therefore, robots in R_2, which are the asleep robots at depth i, are awakened during the same stage. Equality of the sets R_1 and R_2 is shown in (2).

$$|R_2| = |R - R_1| = |R| - \left(\left\lfloor \frac{|R|}{2} \right\rfloor - 1\right) \le \left\lceil \frac{|R|}{2} \right\rceil = |R_1| + 1 \qquad (2$$

For awakening stages according to the algorithm, see the appendix.
The awaking time of set R can be obtained as follows:

- When the depth i is odd, considering that number of robots R_2 are not more than the number of awakened robots $R_1 \cup \{v_r\}$, all of them can be awakened, in parallel and simultaneously. The required time for awakening is equal to the maximum diameter of the rectangle encompassing R_1 and R_2, which is not more than $\frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}} d$.
- When i is even, each robot in R_2 should be simply awakened by one of the robots in $R_1 \cup \{v_r\}$. The required time for awakening is equal to the maximum diameter of the rectangle encompassing R_1 and R_2, which is not more than $\frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}} d$.

Consequently, the required time for awakening the asleep robots in R_2 by the awake ones in R_1 at depth i is:

$$makespan(n, d, i) \le \begin{cases} \left(\frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}}\right) d + makespan\left(\frac{n}{2}, d, i - 1\right) & if\ i\ is\ odd \\ \left(\frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}}\right) d + makespan\left(\frac{n}{2}, \frac{d}{2}, i - 1\right) & if\ i\ is\ even \end{cases} \qquad (3$$

$$|R| \ge 4\ ,\ 1 \le i \le depth$$

Figure 3, shows an example of the processes of space division and awakening. At the depth 1, the set R is divided into two sections, and then the section with higher density will be selected as R_1.
Division is repeated up to the depth of 4, where the number of the robots is less than 4. These robots are awakened by the source robot, and since the depth is even, their awakening time is $\frac{\sqrt{5}}{2^{\lfloor log_2 4 \rfloor}} d + d_v$.

The awakened robots along with the source robot (v_r) start to awaken the robots at the depth 3, given that the depth is odd, the awakening time will not be more than $\frac{\sqrt{2}}{2^{\lfloor log_2 3 \rfloor}} d$. This process continues up to the depth 1, and the total awakening time is the sum of awakening time of all stages.

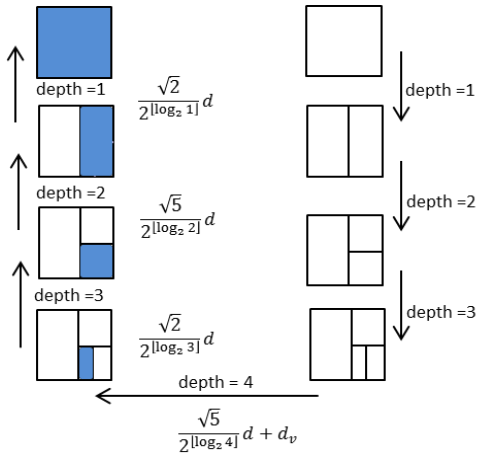


Fig.3. division of the robots space and awakening process for division depth of 4

Theorem: The time required to awaken n asleep robots in a rectangle with sides parallel to the axes of the coordinate system and maximum length and width of d by source robot v_r, given that d_v is the distance between v_r and the farthest asleep robot, is not more than $d_v\left(1 + 2 \times \left(\sqrt{2} + \sqrt{5}\right)\right)$.

Proof: based on (1) and (3), when the depth of division tree is even or odd, we have:

$$makespan(n, d, i) \leq d_v + \begin{cases} \sum_{i\ is\ odd}^{depth} \left(\frac{\sqrt{2}}{2^{\lfloor log_2 i \rfloor}}\right) d \\ \sum_{i\ is\ even}^{depth} \left(\frac{\sqrt{5}}{2^{\lfloor log_2 i \rfloor}}\right) d \end{cases} \quad (4$$

On the other hand $\sum_{i=1}^{n} \left(\frac{1}{2^{\lfloor \log_2 i \rfloor}}\right)$ is a geometric series, thus based on the sum of its n *th* terms and depths of the division tree:

$$\sum_{i=1}^{n} \left(\frac{1}{2^{\lfloor log_2 i \rfloor}}\right) = \sum_{i=0}^{n} \left(\frac{1}{2^i}\right) \approx 2$$

According to the relation (4) and depth of the division tree (both odd and even):

$$makespan(n, d, depth) \leq d_v + 1/2 \times \left(\sqrt{2} + \sqrt{5}\right) \times 2 \times d \leq d_v + \left(\sqrt{2} + \sqrt{5}\right) \times d \quad (5$$

Since $Diam(R) \leq 2d_v$, a circle with diameter of d_v and center point of v_r would encompass all the points of R and the distance between any two points in the circle would be less than or equal to $2d_v$.

$$makespan(n, d, depth) \leq d_v + \left(\sqrt{2} + \sqrt{5}\right) \times d \leq d_v + \left(\sqrt{2} + \sqrt{5}\right) \times 2d_v$$

then

$$makespan(n, d, depth) \leq d_v \left(1 + 2 \times \left(\sqrt{2} + \sqrt{5}\right)\right) = d_v \left(1 + 2 \times \left(\sqrt{2} + \sqrt{5}\right)\right) \leq 8.3 d_v$$

Given that d_v is a lower bound for the awakening time in the optimum condition, the proposed algorithm is an approximation algorithm with an approximation factor of at most 8.3.

4. Time complexity computation

In computing the time complexity of the algorithm, it must be considered that the max, min, and midpoint of a set of n members are obtainable in the time of $O(n)$. Thus, the algorithm can yield an imaginary rectangle encompassing points in R at the time $O(n)$.
The number of the members of R is not less than $n/2$ during each step of implementation of the algorithm on R, this process of calling is repeated until the number of point in the set is less than 4.
Afterward, the algorithm produces some correspondences between the members of R_2 and $R_1 \cup \{v_r\}$; this is done within the time interval of $O(n)$. Therefore, with $T(n)$ as running time of the algorithm for $|R| = n$, we have:

$$\begin{cases} T(n) = T\left(\frac{n}{2}\right) + O(n) & n \geq 4 \\ T(n) = 1 & n < 4 \end{cases} \quad (6$$

The equations (6) yields $T(n) = O(n)$, which means running time of the algorithm is linear.

5. Results

Arkin et al [2] proposed an approximation algorithm with an approximation factor of $O(1)$ and running time of $O(n \log n)$, which ensures a constant approximation factor. However the question is that how big the maximum factor could be? The upper bound of the factors is $\frac{2k-1}{cos(2\pi/k) - sin(2\pi/k)}$ when $k \geq 9$. As can be seen in Figure 4,

the upper bound exceeds 57; in fact, the algorithm does not guarantee any approximation less than 57.

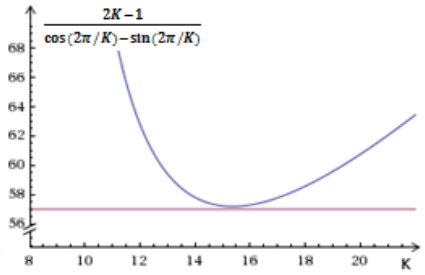


Fig.4. $\frac{2k-1}{cos(2\pi/k)-sin(2\pi/k)}$ [7]

Another disadvantage of the algorithm is its running time. Although, $O(n\,logn)$ is a relatively acceptable running time, it is far from optimum value for FTP problem. This issue is investigated in [7]. Then an approximation algorithm with a constant factor was introduced for Euclidean FTP problem in [7] with a maximum approximation factor of 10.1.
In this study an algorithm with a linear running time which has improved and reduced the approximation factor 8.3.

6. Conclusions

In this paper, an approximation algorithm for solving FTP, using recursively approach was introduced. A better approximation factors, comparing with similar works, was obtained, while the running time was linear. The idea of divide and conquer leads to dividing the asleep robots at each stage into two smaller sets and the set with more density is selected for awakening at the next stage. The division is done while the number of robots in a set is not less than four. By dividing the sets of robots, the algorithm also ensures that number of awaken and asleep robots at every depth are equal (except for the last depth where only one awake robot and maximum three asleep ones participate). As a result, all the asleep robots at the same depth are awakened simultaneously through parallel awakening.

Appendix

Algorithm makespan(R , d , depth)
Input. a set of robot asleep R , depth **=1** and awakeRobotList ← v_r
Output. awakeRobotList

if $R < 4$ *then*
    awakeRobotList ← wakeup set R by awakeRobotList
if $R \geq 4$ *then*
    Find four extreme points, the leftmost, the lowest, the rightmost and the highest of R.
    $Create\ Imaginary\ Rect((x_{min}, y_{min}), (x_{max}, y_{max}))$
    if depth is odd then
        Split R into two subsets with a vertical imaginary line T through the median x-coordinate of the points in R. In case there are multiple points with the same x-coordinate, pick the one with the smallest y-value. Call them $m = (x_{med}, y_{med})$.
            if $((x_{med} - x_{min}) \leq d / 2)$ *then*
                $R_1 = \{(x, y) \in R | (x < x_{med}) or (x = x_{med} \text{ and } y < y_{med})\}$
            else
                $R_1 = \{(x, y) \in R | (x > x_{med}) or (x = x_{med} \text{ and } y > y_{med})\}$
        $R_2 = R - R_1$
        awakeRobotList = makespan(R_1 *, d , depth + 1)*
    else
        Split R into two subsets with a horizontal imaginary line T through the median y-coordinate of the points in R. In case there are multiple points with the same y-coordinate, pick the one with the smallest x-value. Call them $m = (x_{med}, y_{med})$.
            if $((y_{med} - y_{min}) \leq d / 2)$ *then*
                $R_1 = \{(x, y) \in R | (y < y_{med}) \text{ or } (y = y_{med} \text{ and } x < x_{med})\}$
            else
                $R_1 = \{(x, y) \in R | (y > y_{med}) \text{ or } (y = y_{med} \text{ and } x > x_{med})\}$
        $R_2 = R - R_1$
        awakeRobotList = makespan(R_1*, d / 2 , depth + 1)*
    awakeRobotList ← wakeup R_2 *by awakeRobotList, simultaneously*
return awakeRobotList

References

[1] "The open problems project," [Online]. Available: http://maven.smith.edu/~orourke/TOPP/P35.html#Problem.35.

[2] E. Arkin, M. Bender, S. Fekete, J. Mitchell and M. Skutella, "The freeze-tag problem: how to wake up a swarm of robots," *Algorithmica,* vol. 46(2), pp. 193-221, 2006.

[3] D. Bucantanschi, B. Hoffmann, K. R. Hutson and R. M. Kretchmar, "A neighborhood search technique for the freeze tag problem," *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies,* vol. 37, pp. 97-113, 2007.

[4] R. Ravi, "Rapid rumor ramification: approximating the minimum broadcast time," in *In Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, 1994.

[5] A. Bar-Noy, S. Guha, J. Naor and B. Schieber, "Multicasting in heterogeneous networks," in *In Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[6] S. Hedetniemi, S. Hedetniemi and A. Liestma, "A survey of gossiping and broadcasting in communication networks," *Networks,* vol. 18, pp. 319-349, 1988.

[7] Z. Karimi, A. Bagheri and E. Yazdi, "An approximation algorithm with fixed approximation factors and linear running time for Euclidian “freez tag” problem," *16th Computer Society of Iran Computer Conference,* pp. 224-228, 2011.

[8] M. Hammar, B. J. Nilsson and M. Persson, "The online freeze-tag problem," *7th Latin American Symposium,* no. Springer Berlin / Heidelberg, pp. 569-579, 2006.

[9] M. O. Sztainberg, E. Arkin, M. Bender and J. S. B. Mitchell, "Theoretical and experimental analysis of heuristics for the freeze-tag robot awakening problem," *IEEE Transactions on Robotics and Automation,* vol. 20(4), pp. 691-701, 2004.

[10] H. Keshavarz, A. Bagheri, K. Layeghi and S. Mahdavi, "A simulated annealing approach for the freeze-tag problem," in *Recent Trends in Information Systems (ReTIS)*, 2011.

[11] B. Hoffman, "Genetic algorithms applied to the freeze tag problem," in *Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, Denison University, 2004.

[12] B. S. E. Zoraida, M. A. Arock, B. S. M. Ronald and R. Ponalagusamy, "DNA algorithm employing temperature gradient for freeze-tag problem in swarm robotics," *Transactions of the Institute of Measurement and Control,* 2010.

[13] Z. Moezkarimi and A. Bagheri, "A PTAS for geometric 2-FTP," *Information Processing Letters,* vol. 114, no. 12, pp. 670-675, 2014.

[14] K. L. Clarkson, "Approximation algorithms for shortest path motion planning," in *Proceedings of the nineteenthth Annual ACM conference on Theory of Computing (STOC)*, New York, New York, United States, 1987.