Geometric Freeze-Tag Problem

Sharareh Alipour [1,2*], Arash Ahadi [1], Kajal Baghestani [3], Soroush Sahraei [4], Mahdis Mirzaei [4]

[1*]Tehran Institute for Advanced Studies, Tehran, Iran.
[2]Khatam university, Tehran, Iran.
[3]Sharif University of Technology, Tehran, Iran.
[4]University of Tehran, Tehran, Iran.

*Corresponding author(s). E-mail(s): sharareh.alipour@gmail.com;
Contributing authors: aarash.ahadi.academic@gmail.com;
kajal.baghestani83@sharif.edu; soroush.sahraei@ut.ac.ir;
mahdis.mirzaei@ut.ac.ir;

Abstract

The Freeze-Tag Problem (FTP) involves activating a set of initially inactive robots as quickly as possible, starting from a single active robot. Once activated, a robot can assist in activating other robots. Each active robot moves at unit speed. The objective is to minimize the makespan, i.e., the time required to activate the last robot. A key performance measure is the wake-up ratio, defined as the maximum time needed to activate all of the robots in any initial configuration. This work focuses on the geometric (Euclidean) version of FTP in $\mathbb{R}^d$ under the ℓ_p norm, where the initial distance between each inactive robot and the single active robot is at most **1**. For $(\mathbb{R}^2, \ell_2)$, we improve the previous upper bound of **4.62** ([3], CCCG 2024) to **4.31**. The known lower bound for the wake-up ratio is **3.82**. In $\mathbb{R}^3$, we propose a new strategy that achieves a wake-up ratio of **12** for $(\mathbb{R}^3, \ell_1)$ and **12.76** for $(\mathbb{R}^3, \ell_2)$. We also explore the FTP in $(\mathbb{R}^3, \ell_2)$ for specific instances where robots are positioned on a boundary, providing further insights into practical scenarios. Finally, we demonstrate the practical efficiency of our $(\mathbb{R}^2, \ell_2)$ algorithm through simulations on real-world spatial data.

Keywords: Freeze-Tag Problem, Makespan, Approximation Algorithms, Robot Routing, Scheduling, Distributed Computing

---

This work combines and extends results from two previously published papers: Geometric Freeze-Tag Problem [1] (accepted to AAMAS 2025) and Improved Wake-Up Time For Euclidean Freeze-Tag Problem [2] (accepted to CCCG 2025).

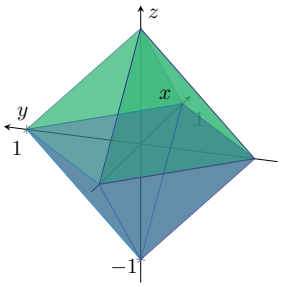


Fig. 1: The unit ball in $(\mathbb{R}^3, \ell_1)$

1 Introduction

The Freeze-Tag Problem (FTP), introduced by Arkin et al. [4], involves activating a set of n initially inactive robots as quickly as possible. The process starts with a single active robot, while the others remain stationary until activated. Once activated, a robot can assist in activating other robots. The objective is to minimize the makespan, which is the time required to activate the last robot.

FTP has practical applications in robotics, particularly in swarm control tasks such as environment exploration [5–8], robot formation [9, 10], and searching [8]. It is also relevant in network design, including broadcast and IP multicast problems [11–13].

This paper examines the Euclidean (geometric) version of FTP. In this version, the input consists of the positions of n inactive robots and one active robot in $\mathbb{R}^d$, along with a distance norm ℓ_p. Each robot's position is represented as a point, and each active robot moves at unit speed. The makespan is defined as the minimum time required to activate the last robot. The wake-up ratio is the maximum makespan over all possible initial configurations of a finite number of robots within a unit ball.

The wake-up ratio is a key metric. If an upper bound c (the wake-up ratio) is known for instances within a unit ball, then for any point set where all inactive robots are within a distance r of the initial active robot, a solution can be found by scaling. This yields a wake-up tree with a makespan of at most $r \cdot c$. Since r is a trivial lower bound on the makespan for that instance, this provides a c-approximation algorithm for the general FTP.

In $\mathbb{R}^2$, the unit ℓ_2-ball is a Euclidean disk, while the unit ℓ_1-ball is a square. In $\mathbb{R}^3$, the unit ℓ_2-ball is a sphere, and the unit ℓ_1-ball is a regular octahedron, as shown in Figure 1.

1.1 Related Work

Studies [14–16] confirm that FTP is NP-hard in $(\mathbb{R}^3, \ell_p)$ for all $p \geq 1$. Similarly, FTP is NP-hard for $(\mathbb{R}^2, \ell_2)$, though its complexity for other norms in $\mathbb{R}^2$ remains an open question [14]. It is conjectured that FTP is also NP-hard for $(\mathbb{R}^2, \ell_1)$ [12].

Let $\gamma_{d,p}$ denote the wake-up ratio in $(\mathbb{R}^d, \ell_p)$. Despite the problem's complexity, few studies have focused on tightening the bounds on this ratio, particularly for the Euclidean plane $(\mathbb{R}^2, \ell_2)$. An early bound of $5\sqrt{2}$ was established, derived from the

result for the ℓ_1 norm ($\gamma_{2,1} \leq 5$ [17]). Most recently, this was improved to 4.62 by Bonichon et al. [3]. These efforts aim to close the gap with the known lower bound of $\gamma_{2,2} \geq 1 + 2\sqrt{2} \approx 3.828$. Consider an instance for FTP in $(\mathbb{R}^2, \ell_2)$ containing exactly 4 robots located at $(0, 1)$, $(0, -1)$, $(1, 0)$, and $(-1, 0)$. It is easy to see that the makespan of this instance is $1 + 2\sqrt{2} \approx 3.83$. In [17] it is conjectured that this input takes the maximum makespan among all instances.

Arkin et al. [4] formulated FTP as the problem of constructing a rooted spanning tree with minimized weighted depth. The root represents the initially active robot, which has a single child, while the n inactive robots serve as nodes having at most two children each. Edges are weighted by the distances in the metric space. This structure is known as the wake-up tree; we define the wake-up time as its weighted depth. In Subsection 3.1 we leverage this spanning tree.

2 Results

For the Freeze-Tag problem in $(\mathbb{R}^2, \ell_2)$, we adapt and extend the crown strategy from [3]. Our approach involves a more detailed case analysis based on the positions of the three inactive robots nearest to the origin. For any given instance, our algorithm selects the strategy from this analysis that yields the minimum makespan.

We present our main results in the following sections. In Section 3, we establish our new upper bound for $\gamma_{2,2}$. In Section 4, we prove an upper bound of 12 for $\gamma_{3,1}$ and 12.76 for $\gamma_{3,2}$.

Finally, we examine a variant of FTP in $(\mathbb{R}^3, \ell_2)$, where the inactive robots are located on the boundary of the unit ℓ_2-ball. We introduce a novel framework that utilizes the results for FTP in $(\mathbb{R}^2, \ell_2)$. Using this framework, we show that this boundary instance can be solved with a makespan of at most 9.4. Formally:

Theorem 2.1 *A robot at the origin can activate any set of n inactive robots on the boundary of the unit ℓ_2-ball in $\mathbb{R}^3$ with a makespan of at most* 9.4.

The proof of this theorem is deferred to Section 5.2. Our approach involves mapping the points on the boundary of the unit ℓ_2-ball in $\mathbb{R}^3$ to an ℓ_2-disk in $\mathbb{R}^2$. By solving the problem in $\mathbb{R}^2$, we obtain a wake-up strategy for $\mathbb{R}^3$. One motivation for studying this setting is our conjecture that the worst-case configuration for FTP occurs when all points lie on the boundary of the unit ball.

A similar wake-up strategy allows us to tackle another variant, which we call surface-FTP. In this version, all robots (both initial active and inactive) are positioned on the surface of a unit ℓ_2-ball in $\mathbb{R}^3$. The distance between two points v and u is their geodesic distance, i.e., the length of the shortest arc connecting them on the surface. This problem has practical applications, such as modeling communication or transport networks on a spherical body like the Earth. Formally:

Theorem 2.2 *Given an instance of surface-FTP, the makespan is at most* 9.92.

This theorem is proved in Section 5.3.

3 FTP in $(\mathbb{R}^2, \ell_2)$

In this section, we show that 4.31 is an upper bound for the wake-up ratio of the FTP in $(\mathbb{R}^2, \ell_2)$.

Theorem 3.1 $\gamma_{2,2} \leq 4.31$.

In Subsection 3.1, we present several algorithms for solving the Freeze-Tag Problem. For any given FTP instance, we select the algorithm that yields the smallest makespan among the proposed options. We then use a computer program to compute the makespan across a range of input instances. Although the space of possible inputs for FTP is infinite, our program evaluates only a finite subset. However, we carefully select representative cases such that every arbitrary input has a makespan close to that of one of the tested instances. As a result, the program introduces a bounded approximation error. In Appendix **B**, we provide a detailed analysis of this approximation error.

We begin by introducing some concepts and notations. Consider a Freeze-Tag instance with n inactive robots, denoted $p_1, \ldots, p_n$, and a single active robot p_0 located at the origin in $(\mathbb{R}^2, \ell_2)$. Let $r_i = |p_i - O|_2$ denote the Euclidean distance from p_i to the origin O, with the assumption that $r_1 \leq r_2 \leq \cdots \leq r_n$. A circle $C(r)$ refers to the circle centered at O with radius r.

We define a crown, denoted by $\mathcal{R}(1-r, \theta)$, as a sector-shaped region bounded by an inner radius r, an outer radius 1, and a central angle θ (see Figure 2). It consists of all points p_i such that $r \leq |p_i - O|_2 \leq 1$ and whose angular coordinates fall within the sector defined by θ. The width of the crown, denoted by w, is given by $w = 1 - r$.

We use $|a - b|$ to denote the absolute value of the difference between a and b. Throughout this paper, we use p_i to refer both to the robot and its initial position.

3.1 Algorithms

We present several algorithms, each of which outperforms the others in specific instances. The algorithms are all based on the crown strategy and, in our analysis, we use the following lemmas from [3]. Let ϕ denote the golden ratio, defined as $\frac{1+\sqrt{5}}{2}$.

Lemma 1. *(Corollary 1 of [3]) There exists a strategy to activate all of the robots in a crown of angle Θ and width w starting with one active robot at a corner in time at most $\Theta + \phi w$.*

Lemma 2. *(Lemma 5 of [3]) There exists a strategy to activate all of the robots in a crown of angle Θ and width w starting with two active robots at a corner on the exterior side of the crown in time at most $\Theta + \left(\frac{\phi^4}{\phi^3 + \Theta}\right) w$.*

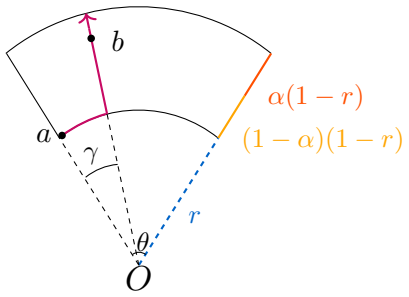


Fig. 2: Activating a crown by one activated robot on the inner corner

Lemma 3 is an extension of Corollary 3 from [3]. Their proof relies on Proposition 14 from [17] and applies only to angles less than or equal to π. In Lemma 3, we demonstrate that the statement holds for any arbitrary angle. The proof is in Appendix **A**.

Lemma 3. *There exists a strategy to activate all of the robots in a crown of angle* Θ *and width* w *starting with one active robot at a corner on the interior side of the crown in time at most* $\Theta + \left(1 + \frac{\phi^4}{\phi^3+\Theta}\right)w$.

First, using a small improvement of the algorithm of [3], we improve the bound of 4.62 to 4.54. Next, by some other methods, we decrease this value to 4.31.

A Small Improvement of the Algorithm of [3]

In [3], the bound 4.62 has been obtained by a trade-off between two strategies. In Strategy 1, p_0 first moves toward p_1. Once p_1 is activated, both p_0 and p_1 move to the origin O. Next, p_0 moves toward p_2 and activates it, while p_1 moves to a point t in the unit circle such that $\angle p_2Ot = \frac{2}{3}\pi$ and $\|O - t\| = r_2$. Finally, each of p_0, p_1, and p_2 activates a crown $\mathcal{R}(1 - r_2, \frac{2}{3}\pi)$.

Now, we improve this strategy. Our improved strategy is denoted by Three-Crowns. Note that the step of coming back to the origin O can be improved. In fact, p_0 can move directly to p_2, and also p_1 can move directly to t. Next, each of p_0, p_1, and p_2 activates a crown. However, we must compute the optimal angle for the crowns that should be activated by these three points.

Let the angles of each of $\mathcal{R}_0$ and $\mathcal{R}_2$ be x; so the angle of $\mathcal{R}_1$ is $2\pi - 2x$. By Lemma 3, the total time of activating $\mathcal{R}_0$ and $\mathcal{R}_2$ is

$$T_{0,2} = r_1 + ||p_1 - p_2||_2 + x + \left(1 + \frac{\phi^4}{\phi^3 + x}\right)(1 - r_2).$$

Also, by Lemma 3, the total time of activating $\mathcal{R}_1$ is

$$T_1 = r_1 + ||p_1 - t||_2 + (2\pi - 2x) + \Big(1 + \frac{\phi^4}{\phi^3 + 2\pi - 2x}\Big)(1 - r_2).$$

We set x such that the maximum of $T_{0,2}$ and T_1 takes the minimum possible value (in this case, $T_{0,2} = T_1$). Thus, the Three-Crowns algorithm activates all robots in at most

$$\tau_1 := \min_{0 \le x \le \pi} \max\{T_{0,2}, T_1\}.$$

In Strategy 2 of [3], first p_0 moves to p_1; next, p_0 starts to activate a crown $\mathcal{R}_0 = \mathcal{R}(1 - r_1, \pi)$ and p_1 starts to activate a crown $\mathcal{R}_1 = \mathcal{R}(1 - r_1, \pi)$, such that the union of $\mathcal{R}_0, \mathcal{R}_1$, and $C(r_1)$ is the unit disk. Since there is no inactive robot in $C(r_1)$, this strategy activates all robots. By Lemma 3, the total time of this strategy is

$$\tau_2 := r_1 + \pi + \Big(1 + \frac{\phi^4}{\phi^3 + \pi}\Big)(1 - r_1).$$

We call this strategy the Two-Crowns algorithm.

The wake-up ratio of 4.62 is found by solving a max-min optimization. For each FTP input, the makespan is calculated using two strategies, and the one with the smallest makespan is chosen. The input with the largest makespan has a value of 4.62. However, by using the Three-Crowns algorithm (a modified version of Strategy 1) and the Two-Crowns algorithm (which is the same as Strategy 2), we can achieve a wake-up ratio of $\min\{\tau_1, \tau_2\} = 4.54$, where the minimum is given over all values of $0 \le r_1 \le r_2 \le 1$.

Two-Crowns algorithm with respect to r_3

In the Two-Crowns algorithm, the active robot p_0 at the origin first moves toward p_1. Once p_1 is activated, there are now two active robots at its position. These robots then move $r_3 - r_1$ units in the direction of $\overrightarrow{Op_1}$. Next, the crown $\mathcal{R}(1 - r_3, 2\pi)$ is divided into two crowns:

$$\mathcal{R}_{\text{left}} = \mathcal{R}(1 - r_3, 2\pi - x) \quad \text{and} \quad \mathcal{R}_{\text{right}} = \mathcal{R}(1 - r_3, x),$$

both starting from the radius that contains p_1. The value of x is determined later. One of these crowns is activated by p_0 and the other by p_1. Since there are no robots, except p_2, inside the circle $C(r_3)$, activating p_2 and both $\mathcal{R}_{\text{left}}$ and $\mathcal{R}_{\text{right}}$ activates the entire unit disk.

Without loss of generality, suppose that robot p_2 is on the right side of $\overrightarrow{Op_1}$. See Figure 3. Consider an instance $p_1, p_2^*, p_3, \ldots, p_n$ of the FTP, similar to $p_1, p_2, \ldots, p_n$, with one difference: the points O, p_2, and p_2^* are collinear, and also $\|O - p_2^*\|_2 = r_3$. By Lemma 3, the crown $\mathcal{R}_{\text{right}}$ can be activated in

$$x + \left(1 + \frac{\phi^4}{\phi^3 + x}\right)(1 - r_3)$$

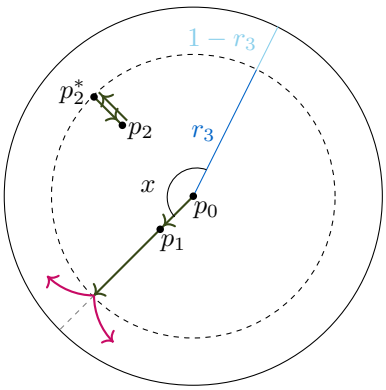


Fig. 3: Two-crowns algorithm with respect to r_3

time units. We now present a strategy to activate p_2 and all robots except p_2^* in $\mathcal{R}_{\text{right}}$. Consider the wake-up tree for activating the crown $\mathcal{R}_{\text{right}}$ by the strategy of Lemma 3; the parent of p_2^* activates p_2. For more details, the parent of p_2^* first moves to the position of p_2^*, then moves to p_2, and finally both p_2 and the parent move back to the position of p_2^*. Other nodes and edges of the wake-up tree do not change.

Since $\|p_2^* - p_2\|_2 = r_3 - r_2$, this strategy takes

$$T_{\text{right}} = x + \left(1 + \frac{\phi^4}{\phi^3 + x}\right)(1 - r_3) + 2(r_3 - r_2)$$

time units.

Activating $\mathcal{R}_{\text{left}}$ by p_1 takes

$$T_{\text{left}} = (2\pi - x) + \left(1 + \frac{\phi^4}{\phi^3 + 2\pi - x}\right)(1 - r_3)$$

time units. Also, moving p_0 and p_1 to a corner of their respective crowns takes r_3 time units.

We set the angle x such that the maximum of T_{right} and T_{left} is minimized (in this case, $T_{\text{right}} = T_{\text{left}}$). Consequently, the total time of the algorithm is

$$r_3 + \min_{0 \le x \le \pi} \max\{T_{\text{right}}, T_{\text{left}}\}.$$

Three-Crowns algorithm with respect to r_3

In the Three-Crowns algorithm, p_0 first moves toward p_1. Once p_1 is activated, there are two active robots at its position. Next, p_1 moves toward p_2 and activates it. Then, both p_1 and p_2 move outward along $\overrightarrow{Op_2}$ for $r_3 - r_2$ units.

At this point, each of p_1 and p_2 activates a crown: p_1 activates a crown $\mathcal{R}_1 = \mathcal{R}(1 - r_3, x)$ and p_2 activates a crown $\mathcal{R}_2 = \mathcal{R}(1 - r_3, x)$, where the value of x will be determined later.

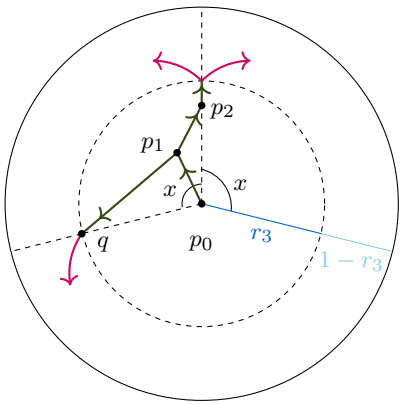


Fig. 4: Three-crowns algorithm with respect to r_3

Meanwhile, p_0 continues to activate robots in the remaining crown $\mathcal{R}(1-r_3, 2\pi-2x)$ as follows. Let q be the point such that $\angle p_2Oq = x$ and $\|O-q\|_2 = r_3$. In this case, p_0 moves toward q; next, p_0 activates a crown $\mathcal{R}_0 = \mathcal{R}(1-r_3, \pi-2x)$. See Figure 4.

By Lemma 3, the total time of activating $\mathcal{R}_1$ and $\mathcal{R}_2$ is

$$T_{1,2} = r_1 + \|p_1 - p_2\|_2 + (r_3 - r_2) + \left(x + \left(1 + \frac{\phi^4}{\phi^3 + x}\right)(1 - r_3)\right).$$

Also the total time of activating $\mathcal{R}_0$ is

$$T_0 = r_1 + \|p_1 - q\|_2 + \Big((2\pi - 2x) + \Big(1 + \frac{\phi^4}{\phi^3 + 2\pi - 2x}\Big)(1 - r_3)\Big).$$

We set the value x such that $T_{1,2} = T_0$. Consequently, the total time of the three-crowns algorithm is

$$\min_{0 \le x \le \pi} \max\{T_{1,2}, T_0\}.$$

(Two or Four)-Crowns Algorithm

First, we outline the overall structure of the Four-Crowns algorithm. In this algorithm, p_0 activates a robot q. Then, both p_0 and q activate new robots, denoted as p' and q', respectively. Finally, the remaining ring-shaped area is divided into the crowns $\mathcal{R}_{p_0}$, $\mathcal{R}_{p'}$, $\mathcal{R}_q$, and $\mathcal{R}_{q'}$, each assigned to one of p_0, p', q, and q'. Since p_0 and p' start their crowns simultaneously, the angles of $\mathcal{R}_{p_0}$ and $\mathcal{R}_{p'}$ should be equal in the optimal case. However, if $\angle p'Oq' = \pi - \beta$ for some $\beta > 0$, then before activating the crowns, some

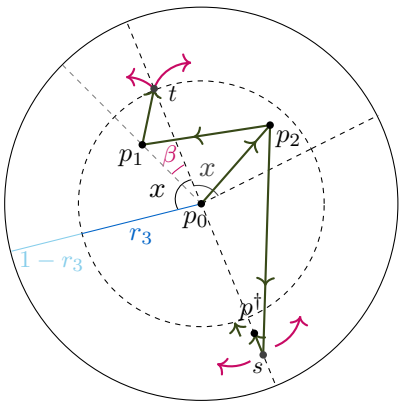


Fig. 5: Four crowns are activated according to the four-crowns algorithm

of those 4 robots must move to new positions p'' and q'' such that $\angle p''Oq'' = \pi$. After this adjustment, they can begin activating their respective crowns.

Let $p^\dagger$ be the robot that is farthest in angular distance from p_1, with the angle $\angle p_1 O p^\dagger = \pi - \beta$ for some β. As discussed, if β is small, the Four-Crowns algorithm may achieve an efficient wake-up time; however, this is not the case when β is large. In such cases, the Two-Crowns algorithm is more efficient. Specifically, it is sufficient to remove the empty region corresponding to the angle β from both crowns. In other words, the Two-Crowns algorithm would activate two crowns of angle $\pi - \beta$ instead of two crowns of angle π. Therefore, based on the value of β, we decide which algorithm to use. These strategies are discussed in detail in the next two parts.

Four-Crowns Algorithm with Respect to β

In this algorithm, first p_0 moves to p_2. Next, p_0 moves to p_1, and p_2 moves to $p^\dagger$, where $p^\dagger$ is the robot that is farthest in angular distance from p_1; let β be an angle such that $\angle p_1 O p^\dagger = \pi - \beta$.

Let t be a point where $\angle p^\dagger O t = \pi$ and $\|t - O\|_2 = r_3$. In the next step, both p_0 and p_1 move toward t to activate their crowns, $\mathcal{R}_0 = \mathcal{R}(1 - r_3, x)$ and $\mathcal{R}_1 = \mathcal{R}(1 - r_3, x)$, respectively.

In order to activate the two other crowns, $\mathcal{R}_2 = \mathcal{R}(1 - r_3, \pi - x)$ and $\mathcal{R}_\dagger = \mathcal{R}(1 - r_3, \pi - x)$ (those of p_2 and $p^\dagger$), note that if the robot $p^\dagger$ is positioned near the inner arc of $\mathcal{R}_\dagger$, it is more efficient for p_2 and $p^\dagger$ to start their crowns from the inner corner rather than from the outer corner. That is, we consider the corner of these two crowns which is the nearest one to $p^\dagger$. In the worst case, $p^\dagger$ is exactly in the middle, so we consider this case and first go to the middle point s and then continue to the corner which is closer to $p^\dagger$ (this also activates $p^\dagger$).

In other words, s is a point on the radius $Op^\dagger$, with a distance of $r_3 + \frac{1-r_3}{2}$ from O. Next, p_2 moves to $p^\dagger$ in the direction of $\overrightarrow{Op^\dagger}$ or $\overleftarrow{Op^\dagger}$. Afterward, both p_2 and $p^\dagger$ continue along this direction until they reach a corner of their crowns. At this point, p_2 and $p^\dagger$ begin activating their crowns. See Figure 5.

Here, we compute an upper bound for the makespan of the Four-Crowns algorithm. By Lemma 3, activating $\mathcal{R}_0$ and $\mathcal{R}_1$ is possible in

$$T_{0,1} = r_2 + \|p_2 - p_1\|_2 + \|p_1 - t\|_2 + \Big(x + \Big(1 + \frac{\phi^4}{\phi^3 + x}\Big)(1 - r_3)\Big)$$

time units. Also, by Lemma 1, activating $\mathcal{R}_2$ and $\mathcal{R}_\dagger$ is possible in

$$T_{2,\dagger} = r_2 + \|p_2 - s\|_2 + \frac{1 - r_3}{2} + (\pi - x) + (1 + \phi)(1 - r_3)$$

time units; we recall that $\phi = \frac{1+\sqrt{5}}{2}$. As with the previous algorithms, we set the angle x such that $T_{0,1} = T_{2,\dagger}$. In other words, the wake-up time of the Four-Crowns algorithm is

$$\min_{0 \le x \le \pi} \max\{T_{0,1}, T_{2,\dagger}\}.$$

Two-Crowns Algorithm with Respect to β

This algorithm is similar to the second strategy in [3]; the only difference is that both crowns have an angle of $\pi - \beta$ (instead of π). Therefore, the total time of this algorithm is

$$r_1 + (\pi - \beta) + \Big(1 + \frac{\phi^4}{\phi^3 + \pi - \beta}\Big)(1 - r_1).$$

Consequently, we consider both the Four-Crowns and Two-Crowns algorithms and select the one with the shorter wake-up time.

Algorithms Activating p_3 Before the Crowns

In some of the previous algorithms, the width of the crowns is $1 - r_3$. Therefore, if r_3 is close to 1, activating the crowns becomes fast. Here, we present some similar algorithms that work better for small values of r_3.

Note that if r_3 is small, then r_1 and r_2 will also be small. As a result, we can apply new algorithms similar to the Three-Crowns and Four-Crowns algorithms.

For more details, as a Four-Crowns type algorithm, p_0 can first activate one of p_1, p_2, or p_3. Then, each of the two active robots activates one of the two remaining inactive robots among the set $\{p_1, p_2, p_3\}$. Let Ω be the angle between these two newly activated robots with respect to the origin O. In the next step, the four active robots reposition themselves by moving along an arc of angle $\frac{1}{2}|\pi - \Omega|$ to arrive at two points on the circle $C(r_3)$. Finally, each of the four robots begins to activate a crown; two of them which are in the same position activate crowns with the angle x, and the other two activate crowns with the angle $\pi - x$.

Depending on which robot is activated first by p_0, the previous paragraph yields three distinct algorithms. The fastest one is determined by the initial positions of p_1, p_2, and p_3.

As a Three-Crowns type algorithm, p_0 can first move to one of p_1, p_2, or p_3. Next, a strategy similar to the Three-Crowns algorithm is employed using one of the two remaining inactive robots. This gives six possible algorithms (three choices for the first activation, two for the second). Note that in each of these six algorithms, the width of the crowns is $1 - r_i$, where $\{p_1, p_2, p_3\} - \{p_i\}$ are two activated robots in the three-crowns algorithm.

3.2 The Main Algorithm

For each instance of the Freeze-Tag Problem, we apply all the discussed algorithms and select the one that minimizes the makespan. In Appendix **B**, we compute the wake-up time for this main algorithm.

4 FTP in $\mathbb{R}^3$

We provide upper bounds for the wake-up ratio of FTP in $(\mathbb{R}^3, \ell_1)$ and $(\mathbb{R}^3, \ell_2)$ using a similar approach for both cases.

Given n points inside a unit ℓ_1-ball (or ℓ_2-ball), we propose an algorithm to activate the robots. Let $\mathcal{D}$ be the unit ℓ_1-ball (ℓ_2-ball) in $\mathbb{R}^3$, and define $\mathcal{D}^+$ as the part of $\mathcal{D}$ with positive z-coordinates, with $\mathcal{D}^- = \mathcal{D} \setminus \mathcal{D}^+$. Without loss of generality, we assume $\mathcal{D}^+$ contains at least as many robots as $\mathcal{D}^-$. The algorithm first activates all robots in $\mathcal{D}^+$. Then, each robot in $\mathcal{D}^+$ simultaneously activates at most one robot in $\mathcal{D}^-$. We now present the details for the ℓ_1 and ℓ_2 cases.

4.1 FTP in $(\mathbb{R}^3, \ell_1)$

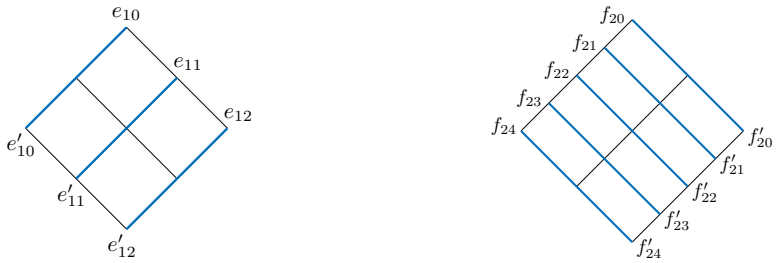
To activate the robots in $\mathcal{D}^+$, the initial active robot p_0 moves to the inactive robot with the smallest z-coordinate, say p_1 (ties are broken arbitrarily). With two active robots now at p_1, we divide $\mathcal{D}^+$ into two regions and assign one to each robot. We will explain in more detail how $\mathcal{D}^+$ is divided. Two active robots move toward the inactive robot with the smallest z-coordinate in their regions. In each step of the algorithm ties are broken arbitrarily. If no inactive robots are left in a region, the assigned robot stops. When a robot activates another in its region, the region splits, and the two active robots continue the process by handling their parts until all robots are activated.

The partitioning works as follows. We consider the projection of the unit ℓ_1-ball onto the $z = 0$ plane, which forms the square $\mathcal{S}$ with corners $(1, 0, 0)$, $(0, 1, 0)$, $(-1, 0, 0)$, and $(0, -1, 0)$. We define a sequence of partitions $\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \ldots$ of this square. Initially, $\mathcal{Q}_0$ consists of the entire square $\mathcal{S}$. At each step, $\mathcal{Q}_i$ is obtained by subdividing each region in $\mathcal{Q}_{i-1}$ into two smaller regions.

For all natural numbers k and i, consider the following points in the plane $z = 0$.

$$e_{ki} = (\frac{i}{2^k}, 1 - \frac{i}{2^k}), \qquad e'_{ki} = (\frac{i}{2^k} - 1, -\frac{i}{2^k}),$$
$$f_{ki} = (-\frac{i}{2^k}, 1 - \frac{i}{2^k}), \qquad f'_{ki} = (1 - \frac{i}{2^k}, -\frac{i}{2^k}),$$

and



(a) $\mathcal{Q}_2$ with 4 squares (b) $\mathcal{Q}_3$ with 8 rectangles

Fig. 6: Two partitions of the square $\mathcal{S}$

$$E_k = \{\overline{e_{ki}e'_{ki}} \,|\, 0 \le i \le 2^k\}, \qquad F_k = \{\overline{f_{ki}f'_{ki}} \,|\, 0 \le i \le 2^k\}.$$

For every integer $t \ge 1$, the partition $\mathcal{Q}_{2t-1}$ divides $\mathcal{S}$ into 2^{2t-1} rectangles using the line segments of $E_{t-1} \cup F_t$. The partition $\mathcal{Q}_{2t}$ then divides $\mathcal{S}$ into 2^{2t} squares using the segments of $E_t \cup F_t$. Therefore, by definition, $\mathcal{Q}_0$ contains the single square $\mathcal{S}$, and $\mathcal{Q}_1$ consists of two rectangles. See Figure 6.

In the first step, p_0 starts at the origin of $\mathcal{D}$ and is responsible for the entire $\mathcal{D}^+$. It moves toward p_1, the inactive robot with the smallest z-coordinate in $\mathcal{D}^+$. Using $\mathcal{Q}_1$, we divide $\mathcal{S}$ into two rectangles. The planes parallel to the z-axis that pass through the segments of E_1 split $\mathcal{D}^+$ into two parts; p_0 takes one, and p_1 takes the other.

In the next step, p_1 moves toward p_2, the inactive robot with the smallest z-coordinate in its assigned part. Once p_2 is activated, we partition the corresponding region using $\mathcal{Q}_1$, splitting it into two parts with planes that pass through segments of E_1 and are parallel to the z-axis. At this time, p_1 is responsible for one part and p_2 for the other. This process repeats until all robots are activated.

We now compute an upper bound for the makespan of the algorithm. Consider the wake-up tree for the points in $\mathcal{D}^+$. Let $p_0 = q_0, q_1, \ldots, q_m$ be an arbitrary path from the root to a leaf in this tree, i.e., in each node of this path there is a robot, either as same as its father or activated by its father. The robot at node q_{i+1} is activated by one of the 2 active robots in its parent node, q_i.

We provide an upper bound for the wake-up time along this path. Let the coordinates of q_i be (x_i, y_i, z_i). We bound the total distance traveled, $\sum_{i=0}^{m-1} |x_i - x_{i+1}| + |y_i - y_{i+1}| + |z_i - z_{i+1}|$. In the algorithm, the z-coordinates are non-decreasing ($z_i \ge z_{i-1}$). Since $z_0 = 0$ and the maximum z-coordinate in $\mathcal{D}^+$ is at most 1, the total vertical travel is bounded:

$$\sum_{i=0}^{m-1} |z_i - z_{i+1}| \le 1. \tag{1}$$

Now, we compute an upper bound for the travel in the xy-plane, $\sum_{i=0}^{m-1} |x_i - x_{i+1}| + |y_i - y_{i+1}|$. Let $a_i = (x_i, y_i)$ be the projection of q_i onto the plane $z = 0$. The distance is then $\|a_i - a_{i+1}\|_1 = |x_i - x_{i+1}| + |y_i - y_{i+1}|$. In the first step, since q_0 is at the origin,

we have $\|a_0 - a_1\|_1 \leq 1$. For $i \geq 1$, when there are two active robots in q_i, then these two robots should activate the region in $\mathcal{Q}_i$ that contains them. So, the diameter of this region is an upper bound for $\|a_i - a_{i+1}\|_1$. If i is even then the diameter of each region of $\mathcal{Q}_i$ is $2^{\frac{2-i}{2}}$ and for odd i, the diameter of $\mathcal{Q}_i$ is $2^{\frac{3-i}{2}}$. This implies:

$$\|a_i - a_{i+1}\|_1 \leq \begin{cases} 2^{\frac{2-i}{2}} & \text{for even } i, \\ 2^{\frac{3-i}{2}} & \text{for odd } i. \end{cases}$$

Therefore:

$$\sum_{i=1}^{m-1} \|a_i - a_{i+1}\|_1 \leq$$
$$1 + (2 + 2 + 1 + 1 + \tfrac{1}{2} + \tfrac{1}{2} + \dots) = 9. \tag{2}$$

So, by (1) and (2), the makespan for activating all robots in $\mathcal{D}^+$ is at most $9+1 = 10$. In the final step, since the number of robots in $\mathcal{D}^+$ is not fewer than in $\mathcal{D}^-$, each robot in $\mathcal{D}^+$ simultaneously activates at most one robot in $\mathcal{D}^-$. This step requires at most 2 time units, as the diameter of $\mathcal{D}$ is 2. Therefore, the total wake-up ratio of FTP in $(\mathbb{R}^3, \ell_1)$ is at most 12.

Theorem 4.1 $\gamma_{3,1} \leq 12$.

4.2 FTP in $(\mathbb{R}^3, \ell_2)$

Our approach for $(\mathbb{R}^3, \ell_2)$ is similar to that for $(\mathbb{R}^3, \ell_1)$. Let $\mathcal{D}$ be the unit ℓ_2-ball in $\mathbb{R}^3$. We define $\mathcal{D}^+$ as the part of $\mathcal{D}$ with positive z-coordinates, and $\mathcal{D}^- = \mathcal{D} \setminus \mathcal{D}^+$. Without loss of generality, assume $\mathcal{D}^+$ contains at least as many robots as $\mathcal{D}^-$.

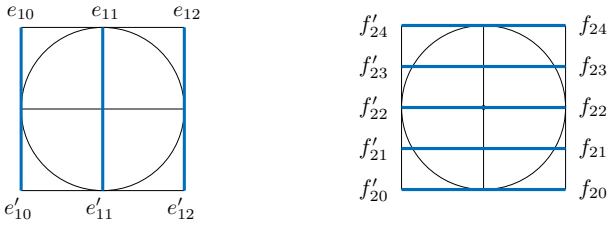
The initial active robot first activates all robots in $\mathcal{D}^+$. Then, each activated robot in $\mathcal{D}^+$ activates a corresponding robot in $\mathcal{D}^-$.

Now, we explain the strategy for activating the robots in $\mathcal{D}^+$. First, p_0 moves toward a point p_1 with the minimum z-coordinate in $\mathcal{D}^+$. Once there are two active robots, $\mathcal{D}^+$ is partitioned into two parts, with each robot responsible for one part. In its assigned region, each robot moves toward the inactive robot with the minimum z-coordinate and activates it. We continue this process until all the robots are activated. The partitioning method is analogous to the one used for $(\mathbb{R}^3, \ell_1)$. For integers $k \geq 0$ and $0 \leq i \leq 2^k$, let:

$$e_{ki} = (-1 + \frac{i}{2^{k-1}}, 1, 0), \qquad e'_{ki} = (-1 + \frac{i}{2^{k-1}}, -1, 0),$$
$$f_{ki} = (1, -1 + \frac{i}{2^{k-1}}, 0), \qquad f'_{ki} = (-1, -1 + \frac{i}{2^{k-1}}, 0),$$

and

$$E_k = \{\overline{e_{ki}e'_{ki}} \,|\, 0 \leq i \leq 2^k\}, \qquad F_k = \{\overline{f_{ki}f'_{ki}} \,|\, 0 \leq i \leq 2^k\}.$$



(a) $\mathcal{Q}_2$ with 4 squares (b) $\mathcal{Q}_3$ with 8 rectangles

Fig. 7: Two partitions of the square $\mathcal{S}'$

Let $\mathcal{S}'$ be a square with the corners $(1,1,0),(-1,1,0),(-1,-1,0)$, and $(1,-1,0)$. We define a sequence of partitions $\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \ldots$ of the unit ℓ_2-disk in the plane $z=0$ as follows.

For every integer $t \geq 1$, the partition $\mathcal{Q}_{2t-1}$ divides $\mathcal{S}'$ into 2^{2t-1} rectangles using the line segments of $E_{t-1} \cup F_t$. The partition $\mathcal{Q}_{2t}$ then divides $\mathcal{S}'$ into 2^{2t} squares using the segments of $E_t \cup F_t$. Therefore, by definition, $\mathcal{Q}_0$ contains the single square $\mathcal{S}'$, and $\mathcal{Q}_1$ consists of two rectangles. See Figure 7.

We now compute an upper bound for the makespan of the algorithm. Consider the wake-up tree for the points in $\mathcal{D}^+$. Let $p_0 = q_0, q_1, \ldots, q_m$ be an arbitrary root-to-leaf path in this tree. At each step of the path, an active robot at node q_i moves to activate the robot at the next node, q_{i+1}.

We provide an upper bound for the total path length, $\sum_{i=0}^{m-1} \|q_{i+1} - q_i\|_2$. Let $q_i = (x_i, y_i, z_i)$. The total distance can be bounded using the triangle inequality:

$$\sum_{i=0}^{m-1} \|q_{i+1} - q_i\|_2 = \sum_{i=0}^{m-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$

$$\leq \sum_{i=0}^{m-1} \left(\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} + |z_i - z_{i+1}| \right).$$

According to our algorithm, the total vertical distance is bounded by $\sum_{i=0}^{m-1} |z_i - z_{i+1}| \leq 1$. Now, let $a_i = (x_i, y_i)$ be the projection of q_i onto the plane $z=0$. So, When q_i moves toward q_{i+1}, the remaining task is to compute $\|a_i - a_{i+1}\|_2$, which gives an upper bound for

$$\sum_{i=0}^{m-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}.$$

Without loss of generality, assume that $a_1 = (x_1, 0)$. In the first step, since q_0 is at the origin, we have $\|a_0 - a_1\|_2 \leq 1$. Next, we show an upper bound for the sum of the next two segments, $\|a_1 - a_2\|_2 + \|a_2 - a_3\|_2$. The maximum for this sum is achieved when the robots are on the boundary of the bounding square $\mathcal{S}'$. Assuming the angle $\angle a_1 a_2 a_3 = \alpha$, the path length is given by $\|a_1 - a_2\|_2 + \|a_2 - a_3\|_2 = 2\sin\frac{\alpha}{2} + 2\sin\frac{3\pi - 2\alpha}{4}$. The maximum value occurs when $\alpha = \frac{3\pi}{4}$. See Figure 8. Thus,

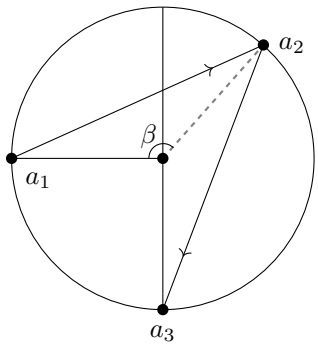


Fig. 8: The maximum value of $\|a_1 - a_2\|_2 + \|a_2 - a_3\|_2$ is achieved when each of the points a_1, a_2, and a_3 is at a distance of 1 from the origin and $\angle a_1 O a_3 = \frac{3}{2}\pi$

$$\|a_1 - a_2\|_2 + \|a_2 - a_3\|_2 \leq 4 \sin \frac{3\pi}{8}. \tag{3}$$

For $i \geq 3$, the distance $\|a_i - a_{i+1}\|_2$ is bounded by the diameter of the region in the partition $\mathcal{Q}_i$ that contains the robots. If i is even, the diameter is $2^{\frac{2-i}{2}}\sqrt{5}$, and for odd i, the diameter is $2^{\frac{3-i}{2}}\sqrt{2}$. This implies:

$$\|a_i - a_{i+1}\|_2 \leq \begin{cases} 2^{\frac{2-i}{2}}\sqrt{5} & \text{for even } i, \\ 2^{\frac{3-i}{2}}\sqrt{2} & \text{for odd } i. \end{cases}$$

Therefore, the sum of these path segments is bounded by the sum of two geometric series:

$$\sum_{i=3}^{m-1} \|a_i - a_{i+1}\|_2 \leq$$

$$\sqrt{2} + \frac{\sqrt{5}}{2} + \frac{\sqrt{2}}{2} + \frac{\sqrt{5}}{4} + \cdots = 2\sqrt{2} + \sqrt{5}. \tag{4}$$

In the final step, each of the active robots in $\mathcal{D}^+$ activates at most one robot in $\mathcal{D}^-$. This step requires at most 2 time units, as the diameter of $\mathcal{D}$ is 2.

Thus, by (3), and (4), the total makespan for activating all robots in $\mathcal{D}$ is at most:

$$1 + 1 + 4 \sin \frac{3\pi}{8} + 2\sqrt{2} + \sqrt{5} + 2 \leq 12.7601.$$

Consequently, the wake-up ratio of FTP in $(\mathbb{R}^3, \ell_2)$ is at most 12.7601.

Theorem 4.2 $\gamma_{3,2} \leq 12.7601$.

Note that our algorithm for computing this upper bound is similar to the one in [18], which gives a bound of 10.06 for FTP in $(\mathbb{R}^2, \ell_2)$. We present an improved analysis; in more detail, we provide a tighter upper bound on the maximum of $\|a_1 - a_2\|_2 + \|a_2 - a_3\|_2$, which helps improve the final result.

5 Wake-up ratio for robots on the boundary of the unit l_2-ball in $\mathbb{R}^3$

In this section, we study FTP in $(\mathbb{R}^3, \ell_2)$ for a special case where the inactive robots are on the boundary of the unit ℓ_2-ball.

We believe this version of FTP is significant, as we hypothesize the maximal makespan occurs when all inactive robots are on the boundary. This hypothesis is based on our observation, that in numerous random instances for various values of n in $(\mathbb{R}^3, \ell_1)$ and $(\mathbb{R}^3, \ell_2)$, the maximum makespan was achieved when the inactive robots were on the boundary.

This scenario is also relevant to real-world applications like communication and transportation, where, similar to the unit ball in $(\mathbb{R}^3, \ell_2)$, key locations on Earth are often on its surface. This leads us to a variant called **surface-FTP**, where all robots, including the initially active one, are located on the surface of the ℓ_2-ball in $\mathbb{R}^3$, and distances are measured using the geodesic (shortest arc) distance.

Our approach for these FTP versions is to map the points on the boundary of the unit ℓ_2-ball in $\mathbb{R}^3$ onto an ℓ_2-disk with radius $\frac{\pi}{2}$. This mapping ensures the Euclidean distance between mapped points is less than their original Euclidean distances on the boundary of l_2-ball.

5.1 Mapping

We begin by cutting the unit ℓ_2-ball in $\mathbb{R}^3$ into two hemispheres and mapping one hemisphere to a disk. For this disk, we generate a wake-up tree using the method from Section 3. This wake-up tree is then used to coordinate the activation process for the robots on the boundary. We explain our mapping below.

Consider the upper hemisphere of a unit ℓ_2-ball with the pole $T = (0, 0, 1)$. Any point $p = (x, y, z)$ on the surface can be uniquely represented by the pair (δ, θ), where $\delta = \arccos(z)$ is the geodesic distance from T to p (the shortest path along the hemisphere), and θ is the angle of the point's projection onto the xy-plane relative to the positive x-axis. This is illustrated in Figure 9.

Mapping $\mathcal{M}$: We define a mapping $\mathcal{M}$ that takes each point p with coordinates (δ, θ) on the hemisphere and maps it to a point p' in a disk of radius $\frac{\pi}{2}$, where p' has polar coordinates (δ, θ). In this mapping, the pole $T = (0, 0, 1)$ is mapped to the origin $(0, 0)$ of the ℓ_2-disk, with $\theta \in [0, 2\pi)$ and $\delta \in [0, \frac{\pi}{2}]$. The boundary of the hemisphere is mapped to a disk of radius $\frac{\pi}{2}$, as shown in Figure 9.

We now prove that for any two points p_1 and p_2 on the hemisphere, the Euclidean distance between them is less than or equal to the Euclidean distance between their mapped points $p_1' = \mathcal{M}(p_1)$ and $p_2' = \mathcal{M}(p_2)$. This is formally stated in the following lemma.

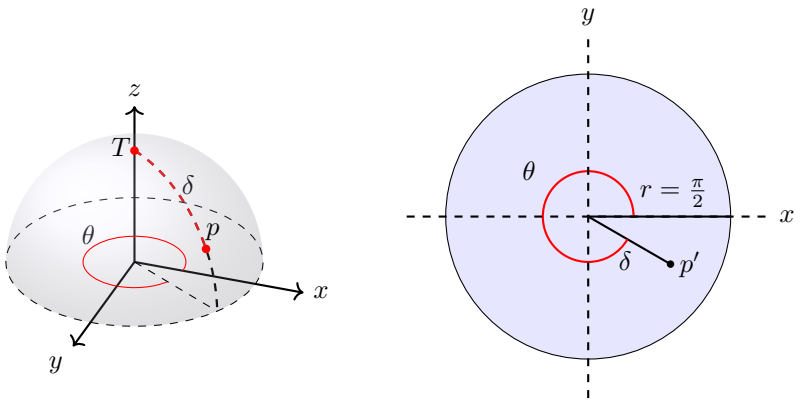


Fig. 9: Every point on the surface of a unit hemisphere can be represented by δ and θ. The point $p = (0.433, 0.75, 0.5)$ is represented by $\theta = -30° = 330°$ and $\delta = 1.047$. The mapping of p onto the unit l_2-disk results in p' with coordinates (δ, θ) in polar coordinates.

Lemma 4. *The mapping $\mathcal{M}$ ensures that for any two points p_1 and p_2 on the hemisphere, mapped to $p'_1 = \mathcal{M}(p_1)$ and $p'_2 = \mathcal{M}(p_2)$ on the disk, $\|p_1 - p_2\|_2 \leq \|p'_1 - p'_2\|_2$.*

Proof We have three cases for $p_1 = (\theta_1, \delta_1)$ and $p_2 = (\theta_2, \delta_2)$.

Case 1: $\theta_1 = \theta_2 = \theta$***, (see Figure 10a)***

In this case, the geodesic distance between p_1 and p_2 is given by $|\delta_1 - \delta_2|$, and $\|p'_1 - p'_2\|_2$ is also equal to $|\delta_1 - \delta_2|$. Since p_1 and p_2 lie on the same meridian, $\|p_1 - p_2\|_2$ is equal to $2\sin\left(\frac{|\delta_1 - \delta_2|}{2}\right)$. Using the inequality $x \leq \sin(x)$ for $x \in \left[0, \frac{\pi}{2}\right)$, we get $\|p_1 - p_2\|_2 \leq \|p'_1 - p'_2\|_2$.

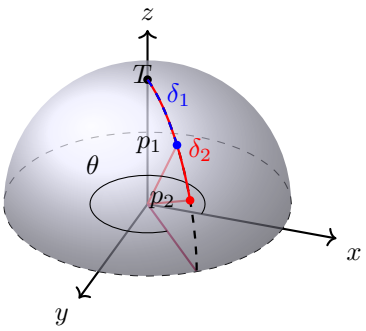
Case 2: $\delta_1 = \delta_2 = \delta$***, (see Figure 10b)***

Consider the circle passing through p_1 and p_2 that is parallel to the xy-plane, with its center on the z-axis. The angle between p_1, the center of the circle, and p_2 is $|\theta_1 - \theta_2|$, and the radius of the circle is $\sin(\delta)$. Consequently, the geodesic distance between p_1 and p_2 is $|\theta_1 - \theta_2| \sin(\delta)$, while $\|p_1 - p_2\|_2$ is $2\sin\left(\frac{|\theta_1 - \theta_2|}{2}\right)\sin(\delta)$, and $\|p'_1 - p'_2\|_2$ is $2\sin\left(\frac{|\theta_1 - \theta_2|}{2}\right)\delta$. Therefore, we have $\|p_1 - p_2\|_2 \leq \|p'_1 - p'_2\|_2$.

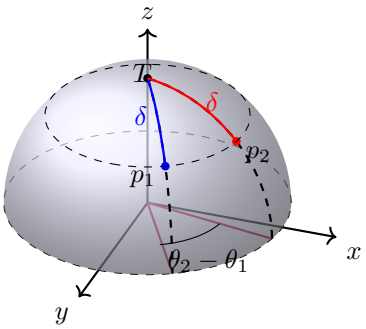
Case 3: $\theta_1 \neq \theta_2$ ***and*** $\delta_1 \neq \delta_2$***, (see Figure 10c)***

Let $q_1 = (\delta_1, \theta_2)$ and $q_2 = (\delta_2, \theta_1)$. Along with p_1 and p_2, these four points form an isosceles trapezoid. And the length of the diagonal of this isosceles trapezoid is equal to $\|p_1 - p_2\|_2$. The first case implies, $\|p_1 - q_2\|_2 = \|p_2 - q_1\|_2 = 2\sin(\frac{|\delta_2 - \delta_1|}{2})$ and the second case implies $\|p_1 - q_1\|_2 = 2\sin(\delta_1)\sin(\frac{|\theta_1 - \theta_2|}{2})$ and $\|p_2 - q_2\|_2 = 2\sin(\delta_2)\sin(\frac{|\theta_1 - \theta_2|}{2})$. Thus,

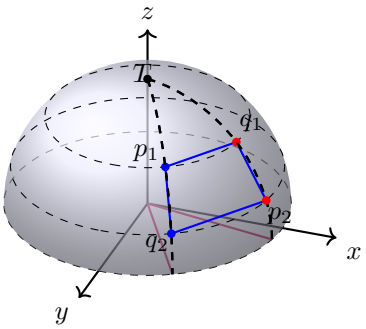
$$\|p_1 - p_2\|_2 = \sqrt{4\sin^2(\frac{|\delta_1 - \delta_2|}{2}) + 4\sin(\delta_1)\sin(\delta_2)\sin^2(\frac{|\theta_1 - \theta_2|}{2})}.$$



(a) In this case, $\theta_1 = \theta_2 = \theta$, and δ_1 and δ_2 represent the geodesic distances of p_1 and p_2 from the point $T = (0, 0, 1)$.



(b) Case 2: In this case, $\delta_1 = \delta_2 = \delta$, where δ is the geodesic distance of both p_1 and p_2 from $T = (0, 0, 1)$.



(c) Case 3: In this case, $\theta_1 \neq \theta_2$ and $\delta_1 \neq \delta_2$. The geodesic distance of p_1 and q_2 from $T = (0, 0, 0)$ is the same, and the geodesic distance of q_1 and p_2 from T is also the same. A plane passes through the points p_1, p_2, q_1, and q_2, forming a trapezoid. The Euclidean distance between p_1 and p_2 is the length of the diameter of this trapezoid.

On the other hand,

$$\|p_1' - p_2'\|_2 = \sqrt{{\delta_1}^2 + {\delta_2}^2 - 2\delta_1\delta_2\cos(|\theta_1 - \theta_2|)}.$$

Now, we show that $\|p_1 - p_2\|_2 \leq \|p_1' - p_2'\|_2$. Consequently we need to show,

$$\sin^2(\frac{|\delta_1 - \delta_2|}{2}) + \sin(\delta_1)\sin(\delta_2)\sin^2(\frac{|\theta_1 - \theta_2|}{2}) \leq (\frac{\delta_1}{2} - \frac{\delta_2}{2})^2 + \frac{\delta_1\delta_2}{2} - \delta_1\delta_2\frac{\cos(|\theta_1 - \theta_2|)}{2}$$

$\sin(x) \leq x$ implies that $\sin^2(\frac{|\delta_1 - \delta_2|}{2}) \leq (\frac{\delta_1}{2} - \frac{\delta_2}{2})^2$. It remains to prove that:

$$\sin(\delta_1)\sin(\delta_2)\sin^2(\frac{|\theta_1 - \theta_2|}{2}) \leq (\delta_1\delta_2)(\frac{1}{2} - \frac{\cos(|\theta_1 - \theta_2|)}{2}).$$

Since $\sin(\delta_1)\sin(\delta_2) \leq \delta_1\delta_2$ and using the identity $2\sin^2(\theta) = 1 - \cos(2\theta)$, the inequality holds, completing the proof.

□

5.2 Strategy for Robots on the Unit Sphere Boundary

For an instance of FTP in $(\mathbb{R}^3, \ell_2)$, where the inactive robots are on the boundary and the initially active robot is at the origin, we propose the following strategy. We split the sphere into two hemispheres. Without loss of generality, assume that the upper hemisphere containing p_1, the robot with the largest z-coordinate, has $n_1 \geq \frac{n}{2}$ robots. We move p_0 toward p_1, activating it at time 1. Then, using the mapping $\mathcal{M}$, we map the robots on the surface of this hemisphere to an ℓ_2-disk with radius $\frac{\pi}{2}$.

We have n_1 inactive robots and two active robots at the boundary of a circle with radius $\frac{\pi}{2} \times r_1$. By definition of p_1, there are no inactive robots inside this circle, so we assign each active robot to activate a crown $\mathcal{R}(\frac{\pi}{2}(1 - r_1), \pi)$. In the worst case, this gives a time to activate $\mathcal{R}(\frac{\pi}{2}, \pi) = \pi + \left(\frac{\phi^4}{\phi^3 + \pi}\right) \times 1 < 3.142 + \left(\frac{6.86}{4.23 + 3.14}\right) < 4.073$ using Lemma 2.

Thus, by Lemma 4 and Lemma 2, after 1 time unit (for moving p_0 to p_1) plus $\frac{\pi}{2} \times 4.073$ time units, all robots in the upper hemisphere are activated, giving us $n_1 + 1$ active robots. Since $n_1 \geq \frac{n}{2}$, each of these newly active robots can activate a robot in the lower hemisphere within 2 time units. Therefore, all robots can be activated in $3 + \frac{\pi}{2} \times 4.073 < 9.4$ time units.

5.3 Strategy for surface-FTP

We apply the mapping approach to solve surface-FTP. Points on the hemisphere's surface are mapped onto a disk with radius $\frac{\pi}{2}$ using the mapping $\mathcal{M}$. In Lemma 4, we proved that the Euclidean distance between two points p_1 and p_2 on the hemisphere is:

$$\|p_1 - p_2\|_2 = \sqrt{4\sin^2(\frac{|\delta_1 - \delta_2|}{2}) + 4\sin(\delta_1)\sin(\delta_2)\sin^2(\frac{|\theta_1 - \theta_2|}{2})}.$$

So, the geodesic distance between them is:

$$2\arcsin\left(\sqrt{\sin^2(\frac{|\delta_1-\delta_2|}{2})+\sin(\delta_1)\sin(\delta_2)\sin^2(\frac{|\theta_1-\theta_2|}{2})}\right).$$

On the other hand, the Euclidean distance between their images $\mathcal{M}(p_1) = p_1'$ and $\mathcal{M}(p_2) = p_2'$ is:

$$\|p_1' - p_2'\|_2 = \sqrt{{\delta_1}^2 + {\delta_2}^2 - 2\delta_1\delta_2\cos(|\theta_1-\theta_2|)}.$$

We claim that the geodesic distance between points p_1 and p_2 is less than or equal to the Euclidean distance between the corresponding points p_1' and p_2'. To support this, we conducted a computational analysis, calculating distances for all combinations of $\delta_1 = \frac{\pi}{2}\epsilon \times i$, $\delta_2 = \frac{\pi}{2}\epsilon \times j$, and $\theta_1 - \theta_2 = \pi\epsilon \times k$, with i, j, k ranging from 0 to $\frac{1}{\epsilon}$. The code, available online[1], was run with $\epsilon = 0.001$. In all tested cases, the geodesic distance was less than or equal to ${\|p_1' - p_2'\|_2}^2$[2]. Thus, the makespan for activating robots on the hemisphere is no greater than the makespan for activating their images on the disk. This means that a wake-up tree for robots on the disk also works for robots on the hemisphere, with a makespan that is no greater.

Now we focus on surface-FTP. Initially, we have an active robot, p_0, on the surface of a sphere, which must activate n inactive robots. Distances are measured as geodesic distances on the surface. We divide the sphere into two halves, with p_0 at the pole of the upper hemisphere.

- We move p_0 to the nearest robot, p_1, which takes ρ_1 time units, where ρ_1 is the geodesic distance. Now, with two active robots, p_1 is tasked with activating robots in the upper hemisphere, while p_0 is tasked with the lower hemisphere.
- **Strategy for the upper hemisphere:** Using the mapping $\mathcal{M}$, we map the points on the upper hemisphere onto a disk of radius $\frac{\pi}{2}$, placing the active robot at $p_1 = (\rho_1, \theta_1)$. We then move p_1 to the center of the disk. The remaining robots on the disk can be activated in $\frac{\pi}{2} \times 4.31$ time units using Theorem 3.1. Therefore, all robots on the upper hemisphere are activated in at most $2\rho_1 + \frac{\pi}{2} \times 4.31$ time units. Since $\rho_1 \leq \frac{\pi}{2}$, the makespan for the upper hemisphere is at most 9.92 time units.
- **Strategy for the lower hemisphere:** First, p_0 moves to p_1 in ρ_1 time units, then to p_1' in $\pi - \rho_1 - \rho_1'$ time units, where p_1' is the nearest point to the lowest point in the hemisphere. Using the mapping $\mathcal{M}$, we map the lower hemisphere onto a disk with radius $\frac{\pi}{2}$. Now, with two active robots, p_0 and p_1', within ρ_1' of the disk's center, the remaining robots can be activated in $\frac{\pi}{2} \times 4.31$ time units using Theorem 3.1. The total makespan for the lower hemisphere is $\pi - \rho_1' + \frac{\pi}{2} \times 4.31 \leq 9.92$.

Thus, the wake-up ratio for surface-FTP is at most 9.92.

[1] Geodesic to Euclidean Distance Ratio Calculator Code
[2] A detailed computation of the derivatives of the distance functions can be found in our code.

6 Experimental Results

We evaluated the performance of our proposed $(\mathbb{R}^2, \ell_2)$ strategies, which are detailed in Appendix **C**, using real-world spatial data from two distinct domains. Our goal was to assess how effectively the algorithm solves instances of the Freeze Tag Problem (FTP) in practical, geographically distributed scenarios.

In the first experiment, we used the locations of pharmacies in New York City, obtained from Google Maps. To normalize spatial relationships, we set Times Square as the origin point $(0, 0)$ and rescaled all pharmacy coordinates so that Euclidean distances would reflect relative travel time or cost. We then simulated a public health emergency, such as the outbreak of COVID-19, where a vital medication is initially available at only one pharmacy. This pharmacy was modeled as the active robot in the FTP framework, while all other pharmacies were inactive. A pharmacy could only receive the medication by contacting one that had already acquired it, reflecting the activation process in the Freeze Tag model. The objective was to minimize the total time needed for the medication to reach all pharmacies. Our algorithm completed this task in approximately 1.225 time units, where the unit is defined by the velocity of movement between pharmacies.

In the second experiment, we applied the same approach to a university campus environment. Specifically, we selected several important and high-traffic locations with spatial data collected from Google Earth, including classrooms, libraries, and faculty buildings. In this scenario, we modeled how urgent information, like a university announcement, spreads from one location to others. As in the pharmacy case, each location becomes informed only when someone from an already-informed location reaches it. Our algorithm completed this task in approximately 2.76 time units.

These results confirm that the $(\mathbb{R}^2, \ell_2)$ strategies detailed in Appendix **C** provide efficient solutions for real-world instances of the Freeze Tag Problem, with applications ranging from emergency resource distribution to fast and reliable information spread in dynamic environments.

Supplementary information. The complete source code and datasets that support the findings of this study are publicly available to ensure reproducibility. These materials, including the programs for all computational analyses and the experimental data from Section 6, can be found in the following repository: repository URL.

Declarations

- Funding: The authors received no financial support for the research, authorship, or publication of this article.
- Competing interests: The authors declare that they have no financial or non-financial interests that are directly or indirectly related to the work submitted for publication.
- Ethics approval: Not applicable.
- Consent to participate: Not applicable.
- Consent for publication: All authors have read and approved the final manuscript for publication.

- Data availability: The datasets generated and analyzed during the current study are publicly available at our repository.
- Materials availability: Not applicable.
- Code availability: The source code used to support the findings of this study is openly available at the following repository: repository URL.
- Author contributions: All authors contributed to the conceptualization and methodology of the study. S. Alipour provided supervision for the project. Investigation and data gathering were performed by K. Baghestani, M. Mirzaei, and S. Sahraei. The original draft was prepared by S. Alipour, A. Ahadi, K. Baghestani, and S. Sahraei. Figures and visualizations were created by A. Ahadi, K. Baghestani, and S. Sahraei. All authors participated in the review and editing of the final manuscript.

References

[1] Alipour, S., Baghestani, K., Mirzaei, M., Sahraei, S.: Geometric freeze-tag problem. In: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems. AAMAS '25, pp. 87–95. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2025). https://dl.acm.org/doi/10.5555/3709347.3743520

[2] Alipour, S., Ahadi, A., Baghestani, K.: Improved Wake-Up Time For Euclidean Freeze-Tag Problem (2025). https://doi.org/10.48550/arXiv.2507.16269

[3] Bonichon, N., Gavoille, C., Hanusse, N., Odak, S.: Euclidean Freeze Tag Problem. In: The Canadian Conference on Computational Geometry (CCCG 2024), St. Catharines, Canada (2024). https://hal.science/hal-04803161

[4] Arkin, E.M., Bender, M.A., Fekete, S.P., Mitchell, J.S.B., Skutella, M.: The freeze-tag problem: how to wake up a swarm of robots. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '02, pp. 568–577. Society for Industrial and Applied Mathematics, USA (2002). https://dl.acm.org/doi/10.5555/545381.545457

[5] Bruckstein, A.M., Mallows, C.L., Wagner, I.A.: Probabilistic pursuits on the grid. The American mathematical monthly **104**(4), 323–343 (1997) https://doi.org/10.1080/00029890.1997.11990644

[6] Gage, D.W.: Minimum-resource distributed navigation and mapping. In: Stein, M.R., Choset, H.M., Gage, D.W., Stein, M.R. (eds.) Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, vol. 4195, pp. 96–103. SPIE, Bellingham, WA (2001). https://doi.org/10.1117/12.417293

[7] Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robotics Autom. **15**(5), 918–933 (1999) https://doi.org/10.1109/70.795795

[8] Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Efficiently searching a graph by a smell-oriented vertex process. Ann. Math. Artif. Intell. **24**(1-4), 211–223 (1998) https://doi.org/10.1023/A:1018957401093

[9] Sugihara, K., Suzuki, I.: Distributed algorithms for formation of geometric patterns with many mobile robots. J. Field Robotics **13**(3), 127–139 (1996)

[10] Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999) https://doi.org/10.1137/S009753979628292X

[11] Arkin, E.M., Bender, M.A., Ge, D.: Improved approximation algorithms for the freeze-tag problem. In: Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures. SPAA '03, pp. 295–303. Association for Computing Machinery, New York, NY, USA (2003). https://doi.org/10.1145/777412.777465

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

[13] Könemann, J., Levin, A., Sinha, A.: Approximating the degree-bounded minimum diameter spanning tree problem. Algorithmica **41**(2), 117–129 (2005) https://doi.org/10.1007/S00453-004-1121-2

[14] Abel, Z., Akitaya, H.A., Yu, J.: Freeze tag awakening in 2d is np-hard. In: 27th Fall Workshop on Computational Geometry, pp. 105–107 (2017)

[15] Johnson, M.P.: Easier hardness for 3d freeze-tag. Proc. 27th Fall Worksh. Comp. Geom.(FWCG) (2017)

[16] Pedrosa, L.L.C., Oliveira Silva, L.: Freeze-tag is np-hard in 3d with l_1 distance. In: Fernandes, C.G., Rajsbaum, S. (eds.) Proceedings of the XII Latin-American Algorithms, Graphs and Optimization Symposium, LAGOS 2023, Huatulco, Mexico, September 18-22, 2023. Procedia Computer Science, vol. 223, pp. 360–366. Elsevier, Netherlands (2023). https://doi.org/10.1016/J.PROCS.2023.08.248

[17] Bonichon, N., Casteigts, A., Gavoille, C., Hanusse, N.: Freeze-Tag in L Has Wake-Up Time Five with Linear Complexity. In: Alistarh, D. (ed.) 38th International Symposium on Distributed Computing (DISC 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 319, pp. 9–1916. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2024). https://doi.org/10.4230/LIPIcs.DISC.2024.9

[18] Yazdi, E.N., Bagheri, A., Moezkarimi, Z., Keshavarz, H.: An o(1)-approximation algorithm for the 2-dimensional geometric freeze-tag problem. Inf. Process. Lett. **115**(6-8), 618–622 (2015) https://doi.org/10.1016/J.IPL.2015.02.011

Appendix

A: Proof of Lemma 3

Let a be the active robot at a corner of the inner arc of the crown. Let b be the robot with the smallest angular separation from a, and let $\gamma = \angle aOb$. First, a moves along the inner arc by an angle of γ. See Figure 2. This takes $T_1 = (1-w)\gamma$ time units. Next, a moves to b, and after activating it, both robots follow the direction $\overrightarrow{Ob}$ until they reach the outer arc of the crown. This takes $T_2 = w$ time units.

By Lemma 2, these two robots can activate the rest of the remaining crown, $\mathcal{R}(w, \Theta - \gamma)$, in $T_3 = (\Theta - \gamma) + \left(\frac{\phi^4}{\phi^3+(\Theta-\gamma)}\right) w$ time units. To prove the lemma, it suffices to show that the total time for these steps is at most $\Theta + \left(1 + \frac{\phi^4}{\phi^3+\Theta}\right) w$. We have:

$$\frac{\phi^4}{\phi^3 + \Theta - \gamma} - \frac{\phi^4}{\phi^3 + \Theta} = \frac{\phi^4\gamma}{(\phi^3 + \Theta - \gamma)(\phi^3 + \Theta)} \leq \frac{\phi^4\gamma}{\phi^3\phi^3} \leq \gamma.$$

So, it holds that:

$$\begin{aligned} T_1 + T_2 + T_3 &= (1-w)\gamma + w + (\Theta - \gamma) + \Big(\frac{\phi^4}{\phi^3 + \Theta - \gamma}\Big)w \\ &= \Theta + \Big(1 - \gamma + \frac{\phi^4}{\phi^3 + \Theta - \gamma}\Big)w \leq \Theta + \Big(1 + \frac{\phi^4}{\phi^3 + \Theta}\Big)w. \end{aligned}$$

B: Computing the Wake-Up Ratio in $(\mathbb{R}^2, \ell_2)$

Due to the mathematical complexity involved in computing the maximum makespan across all our algorithms, we approximated this value using a computational approach.

Consider an input of FTP in $(\mathbb{R}^2, \ell_2)$. The makespans of our algorithms are determined by the parameters r_1, r_2, r_3, and the angles $\mu_{1,2} = \angle p_1 O p_2$, $\mu_{1,3} = \angle p_1 O p_3$, and $\mu_{2,3} = \angle p_2 O p_3$. These six values collectively specify the positions of p_1, p_2, and p_3.

It is important to note that, for the Two-Crowns algorithm, when $r_2 > 0.87$, the resulting makespan is given by:

$$1 + \pi + \left(\frac{\phi^4}{\phi^3 + \pi}\right)(1 - 0.87) \leq 4.27.$$

Thus, we can assume that $r_1 \leq r_2 \leq 0.87$ in our analysis.

Additionally, note that either $\mu_{2,3} = \mu_{1,2} + \mu_{1,3}$ or $\mu_{2,3} = |\mu_{1,2} - \mu_{1,3}|$. We consider both cases for $\mu_{2,3}$ in the computational program.

The performance of the (Two or Four)-Crowns algorithm is influenced by the value of the angle β. Therefore, we consider the maximum makespan over all $\beta \in [0, \pi]$.

In our computer program[3], we divide the interval $[0, 1]$ into 200 equal parts and the angle 2π into 628 equal parts. Specifically, the program computes the makespan for every $r_1, r_2, r_3 \in \left\{0, \frac{1}{200}, \frac{2}{200}, \ldots, 1\right\}$ with $r_1 \leq r_2 \leq r_3$, and every $\mu_{1,2}, \mu_{1,3} \in \{0, 0.01, 0.02, \ldots, 3.14 \approx \pi\}$; see Figure 11(a). For each combination of these parameters, the program computes the minimum makespan among all our algorithms.

The final output of the program is the maximum makespan among all values of $r_1, r_2, r_3, \beta, \mu_{1,2}, \mu_{1,3}$, and $\mu_{2,3}$. The output of our computer program is less than 4.2773. In other words,

$$\max \min(\text{makespan}) \leq 4.2773,$$

where the maximum is taken over all values of $r_1, r_2, r_3, \beta, \mu_{1,2}, \mu_{1,3}$, and $\mu_{2,3}$, and the minimum is taken over all algorithms in Subsection 3.1.

The positions of the robots are arbitrary within the unit ball, but we consider only a finite set of such positions. Consequently, the approximation of 4.2773 introduces some error. We now present an upper bound for this error.

Our intervals divide the unit ball into 31,500 blocks, but only 2×100 of these blocks (corresponding to the first and last intervals of the angles) have intersections. Consider an input of FTP, where p_1, p_2, and p_3 belong to blocks B_1, B_2, and B_3, respectively. By our division, for any block there exists a rectangle with dimensions at most $\frac{1}{100} \times \frac{1}{200}$. Therefore, for each $1 \leq i \leq 3$, there is a corner c_i of block B_i such that $\|p_i - c_i\|_2 \leq \epsilon$, where

$$\epsilon = \frac{1}{2}\sqrt{\left(\frac{1}{100}\right)^2 + \left(\frac{1}{200}\right)^2},$$

See Figure 11(b). The main algorithm is as follows: Apply the computer program to the instance $c_1, c_2, c_3, p_4, p_5, \ldots, p_n$; then, add

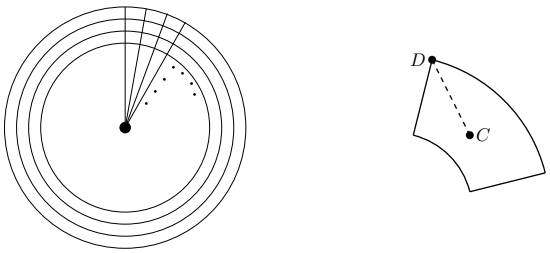
$$2\|p_1 - c_1\|_2 + 2\|p_2 - c_2\|_2 + 2\|p_3 - c_3\|_2$$

to the output (makespan). This value is added because, for each $1 \leq i \leq 3$, to activate c_i a robot q, the robot q can move to c_i, then to p_i, and finally return to c_i. These steps add $\|p_i - c_i\|_2$ times the unit to the makespan.

Consequently, we have

$$2\epsilon_1 + 2\epsilon_2 + 2\epsilon_3 \leq \frac{3\sqrt{5}}{200} \leq 0.0336$$

[3] Available on GitHub



(a) (b)

Fig. 11: (a) dividing a unit disc into 62800 parts, (b) the distance of each point in a part from its closet corner is at most $|CD| = \epsilon$.

as an upper bound for the error of our program. Thus, our total bound is

$$4.2773 + 0.0336 \approx 4.31.$$

C: Two Other Strategies Used in Experimental Results

All algorithms provided in Section 3 are designed based on the position of three points p_1, p_2 and p_3 and then an upper bound to activate a crown by Lemma 3. In this section, we are going to introduce two strategies and then a combined strategy using them, causing an upper bound of 5.4162 for $(\mathbb{R}^2, \ell_2)$. This bound is definitely worse than 4.31, but for special instances the makespan of this combined strategy is better, as it is affected by positions of all robots. Although the upper bound of the combined strategy is greater than the one of Section 3, in our real data instances used in the Experimental Results (Section 6), this strategy provides a better makespan. Note that in the following explanations, the terms p_i are redefined from their meaning in Section 3 and will be defined as needed.

Arc-Strategy

The Arc-Strategy is described as follows for an instance of FTP in $(\mathbb{R}^2, \ell_2)$:

- The active robot, p_0, starts at the origin of a unit ℓ_2-disk and moves to the position of the nearest inactive robot, p_1. At this stage, both p_0 and p_1 are active.
- Next, the disk is divided into two halves by a line passing through the origin and the position of p_1. Each of the two active robots is responsible for activating the inactive robots in one half of the disk. In a parallel step, p_0 activates the nearest inactive robot to the origin in its half, denoted as p_2, while p_1 activates the nearest inactive robot to the origin in its half, denoted as p_3.
- Now, p_0 and p_2 divide their half of the disk into two quarters. p_0 handles its quarter by activating the nearest robot to the origin in its region, while p_2

does the same in its own quarter. Similarly, p_1 and p_3 each take responsibility for their respective quarters, activating the nearest robots to the origin within their areas.

- In each step, when a robot p_i activates another robot p_j, where p_j is the nearest robot to the origin within p_i's assigned portion, p_i and p_j divide their region into two halves. Each robot then takes responsibility for its own half. If p_i is the only robot remaining in its portion, it stops and does nothing in subsequent rounds.

When the Arc-Strategy terminates, all the robots are active. The following lemma provides the makespan for the Arc-Strategy.

Lemma 5. *The Arc-Strategy provides an upper bound of* $3.9651 + 2r_1 + 2r_2 \leq 7.9651$ *for the wake-up ratio in* $(\mathbb{R}^2, \ell_2)$.

Proof At the end of the Arc-Strategy, all robots are active. To determine the wake-up time of the Arc-Strategy, we need to calculate the time it takes to wake up all the robots. The Arc-Strategy produces a wake-up tree, and the wake-up time is the length of the longest path from the center to the leaves of this tree.

Without loss of generality, consider a path $p_0, \ldots, p_k$, which starts at the root, p_0, and ends at a leaf, p_k, in the wake-up tree. We now define the following variables: (see Figure 12).

$$\begin{aligned}
d_i &= \|p_{i-1} - p_i\|_2 \\
r_i &= \|p_i - O\|_2, \text{ where } O \text{ is the center of unit } l_2\text{-disk} \\
C(O, r_i) &= \text{the circle with center } O \text{ and radius } r_i \\
c_i &= \text{cross point of } \overline{p_i O} \text{ and } C(O, r_{i-1}) \\
a_i &= \|p_i - c_i\|_2 \\
b_i &= \|p_{i-1} - c_i\|_2 \\
\alpha_i &= \text{The angle between } \overline{Op_i} \text{ and } \overline{Op_{i+1}} \\
\beta_i &= \text{The angle between } p_{i+1} \text{ and } c_{i+1} \text{ and } p_i
\end{aligned}$$

The total length of the path is expressed as:

$$\sum_{i=1}^{k} d_i = d_1 + d_2 + d_3 + \sum_{i=4}^{k} \sqrt{b_i^2 + a_i^2 - 2a_i b_i \cos\beta_{i-1}}$$

Since $\beta_i = \frac{\pi}{2} + \frac{\alpha_i}{2}$, the path length simplifies to

$$\sum_{i=1}^{k} d_i = d_1 + d_2 + d_3 + \sum_{i=4}^{k} \sqrt{b_i^2 + a_i^2 + 2a_i b_i \sin\frac{\alpha_{i-1}}{2}}$$

Given $b_{i+1} = 2r_i \sin\frac{\alpha_i}{2}$, the total path length becomes:

$$\sum_{i=1}^{k} d_i = d_1 + d_2 + d_3 + \sum_{i=4}^{k} \sqrt{a_i^2 + 4r_{i-1}^2(\sin\frac{\alpha_{i-1}}{2})^2 + 4a_i r_{i-1}(\sin\frac{\alpha_{i-1}}{2})^2}$$

We have,

$$\sum_{i=4}^{k} \sqrt{a_i^2 + 4r_{i-1}^2(\sin\frac{\alpha_{i-1}}{2})^2 + 4a_i r_{i-1}(\sin\frac{\alpha_{i-1}}{2})^2}$$
$$\leq \sum_{i=4}^{k} \sqrt{a_i^2 + 4r_{i-1}^2(\sin\frac{\alpha_{i-1}}{2})^2 + 4a_i r_{i-1}\sin\frac{\alpha_{i-1}}{2}}$$
$$= \sum_{i=4}^{k} \sqrt{(a_i + 2r_{i-1}\sin\frac{\alpha_{i-1}}{2})^2} = \sum_{i=4}^{k} a_i + 2r_{i-1}\sin\frac{\alpha_{i-1}}{2}$$

Now, we have the following inequality:

$$\sum_{i=1}^{k} d_i \ \leq\ d_1 + d_2 + d_3 + \sum_{i=4}^{k} a_i + 2\sum_{i=4}^{k} r_{i-1}\sin\frac{\alpha_{i-1}}{2}$$

Note that $\sum_{i=4}^{k} a_i = 1 - r_3$ and $d_1 = r_1$ and $d_2 \leq r_1 + r_2$ and $d_3 \leq r_2 + r_3$ using triangle inequality so by replacing them and considering all $r_i \leq 1$ we have:

$$\sum_{i=1}^{k} d_i \ \leq\ r_1 + r_1 + r_2 + r_2 + r_3 + 1 - r_3 + 2\sum_{i=3}^{k}\sin\frac{\alpha_i}{2}$$

Given that $\alpha_i \leq \frac{\pi}{2^{i-2}}$ for $i > 1$, we can write:

$$\sum_{i=1}^{k} d_i \ \leq\ 1 + 2r_1 + 2r_2 + 2\left(\sin\frac{\pi}{4} + \sin\frac{\pi}{8} + \dots\right)$$

Using the fact that for $x \geq 0$, $\sin x \leq x$, we have:

$$\sum_{i=1}^{k} d_i \ \leq\ 1 + 2r_1 + 2r_2 + 2\left(\sin\frac{\pi}{4} + \sin\frac{\pi}{8}\right) + 2\pi\left(\frac{1}{16} + \dots + \frac{1}{2^n}\right)$$
$$\sum_{i=1}^{k} d_i \ \leq\ 1 + 2\left(\frac{\sqrt{2}}{2} + 0.3827\right) + 2\pi\left(\frac{1}{16} + \dots\right) + 2r_1 + 2r_2$$
$$\sum_{i=1}^{k} d_i \ \leq\ 1 + (0.7654 + 1.4143) + \frac{\pi}{4} + 2r_1 + 2r_2$$
$$\sum_{i=1}^{k} d_i \ <\ 3.9651 + 2r_1 + 2r_2 \leq 7.9651$$

□

Ring-Strategy

We now introduce the Ring-Strategy. For a given center point, a ring is defined by its inner radius r_1 and outer radius r_2, consisting of all points at a distance r, where $r_1 \leq r \leq r_2$, from the center. Suppose two active robots, p_0 and p_0', are positioned at distance r_1 from the center. We focus on a ring with inner radius r_1 and outer radius 1, and outline a strategy to activate the robots within this ring. The Ring-Strategy is described as follows:

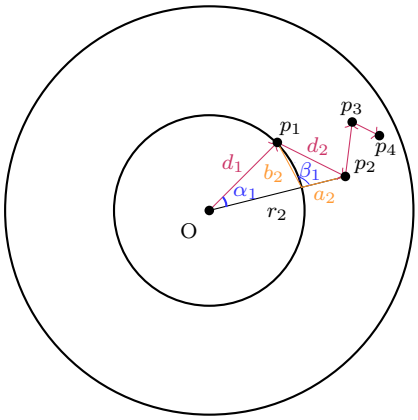


Fig. 12: A path in the Arc-Strategy

- Each of the two active robots is responsible for activating one half of the ring. The ring is split into two equal halves by drawing a diameter through $C(O, r_1)$ that passes through p_0. Next, we explain the strategy for p_0.
- Consider the projections of the points in p_0's half of the ring onto the circle $C(O, r_1)$. The projection of a point p_j onto the circle $C(O, r)$ is the intersection of $\overline{Op_j}$ with the circle $C(O, r)$. Let p_1 be the point whose projection is closest to p_0. Then, p_0 activates p_1.
- In the next step, p_0 and p_1 divide their half of the ring into two smaller half-rings using the circle $C(O, \frac{r_1+1}{2})$. This creates one half-ring with an inner radius of r_1 and an outer radius of $\frac{r_1+1}{2}$, and another with an inner radius of $\frac{r_1+1}{2}$ and an outer radius of 1. Now, each of p_0 and p_1 is responsible for activating robots in their respective half-rings.
- As before, p_0 activates the point in its half-ring whose projection onto $C(O, r_1)$ is closest to it, while p_1 activates the point in its half-ring whose projection onto $C(O, \frac{r_1+1}{2})$ is closest to p_1.
- In each subsequent step, the active robot p_l in a half-ring with inner radius r_i and outer radius r_j identifies the point in its half-ring whose projection onto $C(O, r_i)$ is closest and activates it. The half-ring is then split into two smaller half-rings: one with inner radius r_i and outer radius $\frac{r_i+r_j}{2}$, and the other with inner radius $\frac{r_i+r_j}{2}$ and outer radius r_j. At each step, the thickness of the half-ring is halved, with each robot responsible for activating the robots in their respective half-ring.
- Meanwhile, similar to p_0, p_0' activates the robots in its half of the ring. By the end of this process, all robots within the ring are activated.

Lemma 6. *If in an instance of FTP all the robots are inside a ring with an inner radius of r_1 and an outer radius of 1, and we have two active robots on the boundary of $C(O, r_1)$, then the Ring-Strategy activates all robots in at most $\pi + 3 - 3r_1$ time units.*

Proof Since there are two active robots at the same initial position, p'_0 activates the robots in its own half in parallel. Therefore, the upper bound on the wake-up time is the same for both halves, and all robots will be activated within this time.

Let $p_0, p_1, \ldots, p_k$ represent a path in the wake-up tree of the Ring-Strategy, with the root at p_0. We now define the following variables (see Figure 13):

$$\begin{aligned} d_i &= \|p_{i-1} - p_i\|_2 \\ (r_i, \alpha_i) &= \text{polar coordinates of the location of } p_i \\ a_i &= \text{the distance between } (r_{i-1}, \alpha_{i-1}) \text{ and } (r_{i-1}, \alpha_i) \\ &\quad \text{i.e the distance between } p_i \text{ and image of} \\ &\quad p_{i+1} \text{ on } C(O, r_i) \\ b_i &= \text{the distance between } (r_{i-1}, \alpha_i) \text{ and } (r_i, \alpha_i) \\ &\quad \text{i.e the distance between } p_i \text{ and image of} \\ &\quad p_i \text{ on } C(O, r_{i-1}) \end{aligned}$$

The total length of the path $p_0, p_1, \ldots p_k$ is given by:

$$\sum_{i=1}^{k} d_i \leq \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i.$$

Since we split the half-ring associated with each active point at each step, we have $b_{i+1} \leq \frac{1-r_1}{2^{(i-1)}}$ for $i \geq 1$ and $\sum_{i=1}^{k} a_i \leq \pi$. Thus,

$$\sum_{i=1}^{k} d_i \leq \pi + b_1 + \sum_{i=2}^{k} \frac{1 - r_1}{2^{(i-2)}}$$

$$\leq \pi + 1 - r_1 + 2(1 - r_1) = \pi + 3(1 - r_1)$$

□

Combined Strategy for FTP in $(\mathbb{R}^2, \ell_2)$

We now propose a combined strategy based on the Arc-Strategy and Ring-Strategy. Given an instance of FTP in $(\mathbb{R}^2, \ell_2)$ with an active robot at the origin, let r_1 be the distance to the nearest inactive robot, p_1.

In the first step, p_0 moves to activate p_1. The disk is then divided into two halves by the diameter passing through the position of p_1. Now there are two active robots at this location, and each is responsible for one half of the disk.

Assume we want to activate the robots in the right half using p_0. Let p_2 be the second nearest point to the center in the right half of the ring and r_2 represent

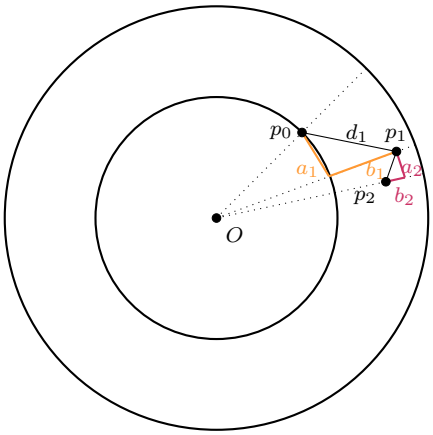


Fig. 13: A path in the Ring-Strategy.

its distance from the center. If $r_2 \leq 0.3627$, then using the Arc-Strategy, this half can be activated in $2r_1 + 2r_2 + 3.9651 \leq 4r_2 + 3.9651 \leq 5.4162$ time units, see the proof of Lemma 5. Otherwise, using the Ring-Strategy, p_0 moves $r_2 - r_1$ units in the direction of $\overrightarrow{Op_1}$. The half-ring on the right now has an inner radius of r_2 and an outer radius of 1. So, we can activate the robots in this half-ring in $r_1 + (r_2 - r_1) + 3 + \pi - 3r_2 < 6.1416 - 2 \times 0.3627 = 5.4162$ time units, see the proof of Lemma 6.

Similarly, let $p_{2'}$ be the second-nearest robot to the origin in the left half of the disk. Based on its distance $r_{2'}$, we choose the better of the two strategies for that half, which also takes at most 5.4162 time units.