Freeze-Tag in L_1 Has Wake-Up Time Five with Linear Complexity

Nicolas Bonichon
LaBRI, University of Bordeaux, CNRS, Bordeaux INP, France

Arnaud Casteigts
LaBRI, University of Bordeaux, CNRS, Bordeaux INP, France
CS Department, University of Geneva, Switzerland

Cyril Gavoille
LaBRI, University of Bordeaux, CNRS, Bordeaux INP, France

Nicolas Hanusse
LaBRI, University of Bordeaux, CNRS, Bordeaux INP, France

Abstract

The Freeze-Tag Problem, introduced in Arkin et al. (SODA'02) consists of waking up a swarm of n robots, starting from a single active robot. In the basic geometric version, every robot is given coordinates in the plane. As soon as a robot is awakened, it can move towards inactive robots to wake them up. The goal is to minimize the makespan of the last robot, the *makespan*.

Despite significant progress on the computational complexity of this problem and on approximation algorithms, the characterization of exact bounds on the makespan remains one of the main open questions. In this paper, we settle this question for the ℓ_1-norm, showing that a makespan of at most $5r$ can always be achieved, where r is the maximum distance between the initial active robot and any sleeping robot. Moreover, a schedule achieving a makespan of at most $5r$ can be computed in time $O(n)$. Both bounds, the time and the makespan are optimal. Our results also imply for the ℓ_2-norm a new upper bound of $5\sqrt{2}r \approx 7.07r$ on the makespan, improving the best known bound of $(5 + 2\sqrt{2} + \sqrt{5})r \approx 10.06r$.

Along the way, we introduce new linear time wake-up strategies, that apply to any norm and show that an optimal bound on the makespan can always be achieved by a schedule computable in linear time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases freeze-tag problem, metric, algorithm

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.9

Related Version *Full Version*: `http://arxiv.org/abs/2402.03258` [7]

Funding The last three authors are supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL).

Acknowledgements We thank J.S. Mitchell and the anonymous referees for their advice on a previous version of this article.

1 Introduction

In a collaborative swarm of robots, individual robots often have limited capacities in terms of energy, sensing, computation, movement or communication. They cooperate in order to achieve global tasks like exploring a network [11] or planning the motion of each individual robot without conflict [24]. As the robots energy is limited, it may be necessary to switch them off and wake them up later, which requires an efficient way to do so.

The Freeze-Tag Problem (FTP) is an optimization problem that consists of activating as fast as possible a swarm of robots represented by points in some metric space (in general, not necessarily Euclidean). Active (or awake) robots can move towards any point of the space at a constant speed, whereas inactive robots are asleep (or frozen) and can be activated only by a robot moving to their position. Initially, there are n sleeping robots and one awake robot. The goal is to determine a schedule whose makespan is minimized; that is, the time until all the robots have been activated is minimized. FTP has applications not only in *robotics*, e.g. with group formation, searching, and recruitment, but also in *network design*, e.g. with broadcast and IP multicast problems. See [4, 3, 18] and references therein.

State of the art. FTP is NP-Hard in high dimension metrics like centroid metrics [3] (based on weighted star n-vertex graphs) or unweighted graph metrics with a robot per node [4]. Many subsequent works have extended this hardness result to constant dimensional metric spaces, including the Euclidian ones. A series of papers [1, 15, 21] proves that FTP is actually NP-Hard in $(\mathbb{R}^3, \ell_p)$, for every $p \geq 1$, i.e., in 3D with any ℓ_p-norm[1]. For 2D spaces, this remains NP-Hard in $L_2 = (\mathbb{R}^2, \ell_2)$, leaving open the question for other norms [1]. It is believed, see [3, Conjecture 28], that FTP remains NP-Hard in $L_1 = (\mathbb{R}^2, \ell_1)$. Beyond being interesting from a theoretical point of view, the L_1 case corresponds to the case where movements are restricted to be orthogonal. This is the case for swarms of robots in certain warehouses.

Several approximation algorithms and heuristics were designed. In their seminal work, [3] developed a 14-approximation for centroid metrics, and a PTAS for $(\mathbb{R}^d, \ell_p)$. The authors of [4] presented a $O(1)$-approximation for unweighted graph metrics with one robot per node, and a greedy strategy analyzed in [25] gives a $O(\log^{1-1/d} n)$-approximation in $(\mathbb{R}^d, \ell_p)$. For general metrics, the best approximation ratio is $O(\sqrt{\log n}\,)$ [18]. For heuristics, several experimental results can be found in [9, 10, 17]. See [5, 20, 14, 8] for generalizations and variants of the problem, including the important online version.

As observed by [2], the FTP can be rephrased as finding a rooted spanning tree on a set of points with minimum depth, where the root node (corresponding to the awake robot) has one child and all the other nodes (corresponding to the n sleeping robots) have at most two children (see Figure 1). Each edge has a non-negative *length*, representing the distance in the metric space between its endpoints. Such a tree is called a *wake-up tree*, and its weighted-depth is called the *makespan*.

This problem can be approached by constructing a bounded-degree B minimum diameter spanning tree (BDST), whose best known approximation is $O(\sqrt{\log_B n})$ [18]. As shown in [3], the BDST problem for $B = 2$ can be approximated within a constant using approximation algorithms for Traveling Salesman Problem, in its *metric* version (hereafter, simply TSP). Moreover, the Path-TSP, a generalization of TSP in which one asks for finding a minimum path length spanning a point set from given start and end points, provides a valid wake-up tree and thus a solution for FTP. The link with TSP is reenforced by the recent approximated reduction of Path-TSP to TSP [26]. In fact, as shown by [18], the BDST problem for $B = 3$, implies the same guarantee for FTP, i.e., $O(\sqrt{\log n})$ times the optimal.

This being said, there are significant differences between TSP and FTP, the latter being considered as a cooperative TSP version where awake robots can help in visiting unvisited cities. First, from an algorithmic point of view, the best lower bound on the approximation

[1] The ℓ_p-norm of a given a vector $u = (u_1, \ldots, u_d) \in \mathbb{R}^d$ is defined by $\ell_p(u) = (\sum_{i=1}^{d} |u_i|^p)^{1/p}$. We denote by $(\mathbb{R}^d, \ell_p)$ the d-dimensional normed linear space where the distance between two points $u, v \in \mathbb{R}^d$ is given by $\ell_p(u - v)$. We denote by L_p the 2D normed linear space $(\mathbb{R}^2, \ell_p)$.

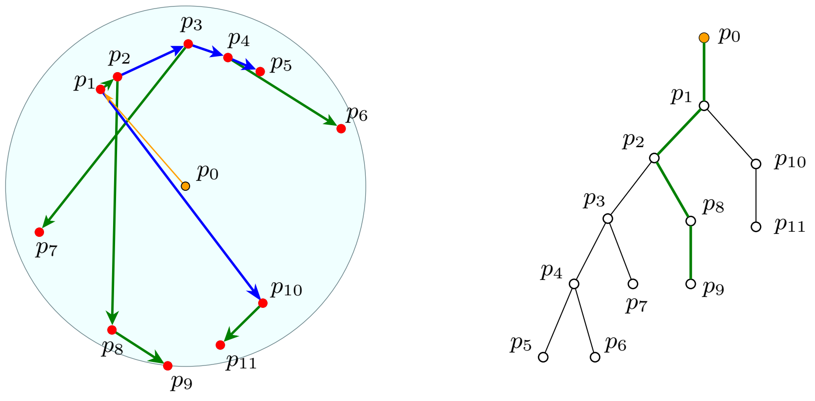

Figure 1 Example of a (here, Euclidean) instance of FTP (on the left). The robot at p_0 must wake up $n = 11$ sleeping robots at $p_1, \ldots, p_n$. In this example, positions are normalized in the unit ℓ_2-disk, p_0 being at the center. An optimal solution, depicted by arrows, can be represented as a binary weighted tree (right). The makespan is the length of the longest (weighted) branch in that tree, here 2.594, corresponding to the path $(p_0, p_1, p_2, p_8, p_9)$. Observe that, even if the sleeping robots are in a convex configuration, the optimal solution may have multiple edge crossings.

factor are $5/3 - \varepsilon$ for FTP [3], and only $123/122 - \varepsilon$ for TSP [16] (assuming $\mathsf{P} \neq \mathsf{NP}$). On the other side, the time complexity for PTAS in $(\mathbb{R}^d, \ell_p)$ is $n(\log n)^{(d/\varepsilon)^{O(d)}}$ for TSP [6, 22, 19] vs. $O(n \log n) + 2^{(d/\varepsilon)^{O(d)}}$ for FTP [3], subject to $\varepsilon \leq \varepsilon_d$, where ε_d depends on the number of dimensions d. Second, and perhaps more fundamentally, it is well known that, even in the unit ball in $(\mathbb{R}^d, \ell_p)$, the shortest spanning path (or tour) has unbounded length in the worst-case (it depends on n), whereas the makespan for FTP is bounded by a constant (that does not depend on n). For TSP, the worst-case length is $\Theta_d(n^{1-1/d})$ [12], whereas for FTP the worst-case optimal makespan is no more than some constant ρ_d, independent of n [3].

The constant ρ_d plays an important role for PTAS and approximation algorithms. For instance, it drives the condition "$\varepsilon \leq \varepsilon_d$" in the grid refinement approach of [3], where local solutions in radius-$(1/\varepsilon)$ balls have to be constructed. For $(\mathbb{R}^2, \ell_2)$, the constant ρ_2 coming from the approach of [3] has been proved to be at most 57 by [27]. The latter authors also construct in time $O(n)$ a wake-up tree of makespan at most $5 + 2\sqrt{2} + \sqrt{5} \approx 10.06$, which is the best known upper bound for ρ_2.

Main contributions. Although some of our results hold for arbitrary norms η, we mostly focus on the ℓ_1-norm assuming that robots are spread in the plane $\mathbb{R}^2$ within a *normalized disk of radius* 1 *centered at its initial awake robot of coordinate* $(0,0)$. Note that the unit ℓ_2-disk is a usual disk whereas the unit ℓ_1-disk is a square, rotated by 45 degrees with respect to the coordinate axes.

Our results are summarized in Table 1 and deal with lower and upper bounds on the makespan for different norms, as well as their algorithmic solutions. For ℓ_p-norms, the upper bounds mainly come from our new upper bound for ℓ_1 (Theorem 1). Moreover, we show how to build, in linear time, a wake-up tree achieving a makespan of at most the best upper bounds known for arbitrary norm (Theorem 2).

More precisely, we get:

▶ **Theorem 1.** *A robot at the origin can wake up any set of n sleeping robots in the unit ℓ_1-disk with a makespan of at most* 5*. The wake-up tree can be constructed in $O(n)$ time.*

Table 1 Lower and upper bounds on the makespan denoyed by the wake-up constant $\gamma(\eta)$ for different norms η. The number $\pi(\eta) \in [3,4]$ (resp. $\Lambda(\eta) \in [2, \pi(\eta)]$) is the half-perimeter of a unit η-disk (resp. of the largest inscribed parallelogram in the unit η-disk), measured in the η-metric. $\varphi = (1+\sqrt{5})/2$ is the golden ratio. Note that by Theorem 2, all our upper bounds are complemented with a linear time algorithm constructing wake-up trees of makespan no more than these bounds.

Norm	Lower bounds (*Theorem 10*)	Upper bounds	References
ℓ_1,ℓ_∞	5	5	*Theorem 1*, Section 3
ℓ_2		$5+2\sqrt{2}+\sqrt{5} \approx 10.06$	[27]
ℓ_2	$1+2\sqrt{2} \approx 3.83$	$5\sqrt{2} \approx 7.07$	*Corollary 13*, Section 5
ℓ_p	$1+2^{1+\max(1/p,1-1/p)}$	$5 \cdot 2^{\min(1/p,1-1/p)}$	*Corollary 13*, Section 5
Arbitrary η	$1+\Lambda(\eta) \in [3, 1+\pi(\eta)] \subseteq [3,5]$	$3+\varphi\pi(\eta) \leq 9.473$	*Corollary 11*, Section 5

Obviously, if the awake robot is at distance at most r from all the sleeping robots, then by scaling the unit disk with their positions, and by using Theorem 1, one can construct a wake-up tree of makespan of at most $5r$. By a loose argument, this yields also a 5-approximation $O(n)$ time algorithm for $(\mathbb{R}^2, \ell_1)$, since r is a trivial lower bound on the makespan. A similar statement holds for $(\mathbb{R}^2, \ell_\infty)$.

Both bounds in Theorem 1 are optimal: the makespan of 5, and obviously the linear time construction of the wake-up tree. The upper bound of 5 is reached with $n = 4$ sleeping robots at positions $(\pm 1, 0)$ and $(0, \pm 1)$.

By a simple argument, Theorem 1 immediately improves the best known upper bound for the ℓ_2-norm. Indeed (see also Corollary 13), by scaling the unit ℓ_2-disk, we can use the construction of Theorem 1 to obtain a makespan of $5\sqrt{2} \approx 7.07$ for the unit ℓ_2-disk, improving upon the previous 10.06 upper bound of [27].

Our second result concerns algorithmic aspects of the FTP. Theorem 2 (and its simplified version in Corollary 12) states that there is a linear time algorithm that can match the best known upper bound to wake up a unit disk. The result is general enough to hold in any normed linear space $(\mathbb{R}^2, \eta)$, a.k.a. Minkowski plane.

To make the statement of Theorem 2 precise, let us define $\gamma_n(\eta)$ as the worst-case optimal makespan of a wake-up tree for any set of n sleeping robots in the unit η-disk and rooted at the origin. In other words, $\gamma_n(\eta)$ is the best possible upper bound of the makespan to wake up n sleeping robots from an awake robot placed at the origin, in the unit η-disk, all distances being measured according to the η-metric. Finally, let us introduce the *wake-up constant* w.r.t. the η-norm defined by

$$\gamma(\eta) \quad = \quad \sup_{n \in \mathbb{N}} \gamma_n(\eta) \ .$$

Note that the constant ρ_2 introduced above is simply $\gamma(\ell_2)$, the ℓ_2 wake-up constant.

▶ **Theorem 2.** *Let η be any norm and let $\tau > 3$ be any real such that $\tau \geq \gamma(\eta)$. In time $O(n)$, a wake-up tree of makespan at most τ, rooted at the origin, can be built for any set of n points in the unit η-disk.*

So, if one plugs $\eta = \ell_1$ and $\tau = 5$ in Theorem 2, then proving that $\gamma(\ell_1) \leq 5$ becomes sufficient to obtain a linear time construction of a wake-up tree of makespan at most 5 as claimed in Theorem 1. In other words, given Theorem 2, the main Theorem 1 can simply be restated as: $\gamma(\ell_1) \leq 5$. Furthermore, as already explained, the bound of 5 is attained for $n = 4$ sleeping robots, so $\gamma(\ell_1) \geq \gamma_4(\ell_1) = 5$, and the wake-up constant in ℓ_1-norm is thus 5.

To prove Theorem 1 and Theorem 2, we need several intermediate results, which we believe are of independent interest.

Table 2 Makespan and complexity to wake-up, from the origin, n sleeping robots in a cone of arc-length w in the unit η-disk.

Strategy	Makespan	Complexity	References
`Heap-Strategy`	$1 + w \log_2 n$	$O(n)$	*Proposition 7*, Section 4.1
`Split-Cone-Strategy`	$1 + \varphi w$	$O(n \log n)$	*Proposition 8*, Section 5.1
`Linear-Split-Strategy`	$1 + \varphi w + O(w \cdot (\log^3 n)/n)$	$O(n)$	*Proposition 9*, Section 5.1

In particular, we show how to build efficiently wake-up trees with a small makespan in subregions of the unit disk, like truncated cones (that is the part of a cone within the unit disk) of arc length w. The algorithms presented in Table 2 are named with respect to their underlying strategies. The complexity and makespan of `Linear-Split-Strategy` come from a non-trivial combination of two strategies: `Heap-Strategy` and `Split-Cone-Strategy`, introduced in this paper.

Outline. The proof of Theorem 1 is divided into two parts: (1) the upper bound is presented in Section 3 and its proof, based on a strategy called `L1-Strategy`, is constructive; (2) the linear complexity of the construction is then proved in Section 4, based on `Heap-Strategy`. Theorem 2 is proved by the use of similar strategies. In Section 5, we describe `Split-Cone-Strategy` and show how these strategies can be combined to get new bounds for arbitrary norms. Due to space limitations, some of the proofs are deferred to the Full Version [7], while providing a summary of the main ideas in the body of the paper.

2 Preliminaries: from cones to triangles and squares

Most of our algorithms are based on a partitioning of the unit disk into several shapes. The basic shape is a *cone* centered at the initial awake robot $p_0 = O$. Given two points A, B of the unit circle centered at the origin O, we denote by $\mathsf{arc}(A, B)$ the part of the circle that is traversed anti-clockwise from A to B. The length of $\mathsf{arc}(A, B)$ is denoted by $|\mathsf{arc}(A, B)|$, and is called *arc-length*. We have $|\mathsf{arc}(A, B)| \in [0, 2\pi(\eta))$, where $\pi(\eta)$ is the half-perimeter of a unit η-disk. Note that arc-length and half-perimeter are measured in the η-norm. For instance, $\pi(\ell_2) = 3.14...$ and $\pi(\ell_1) = 4$. Then, $\mathsf{cone}(A, w)$, where $w = |\mathsf{arc}(A, B)|$, is the region of the unit η-disk bounded by $\mathsf{arc}(A, B)$ and the segments $[OA]$ and $[OB]$.

Let us focus on L_1. The unit ℓ_1-disk has a four-fold symmetry and has perimeter $2\pi(\ell_1) = 8$. We consider two specific cones: *squares* and *triangles*. See Figure 2. More precisely, and up to scaling and symmetry along the four axis, a *square* is a $\mathsf{cone}(P, 2)$ with a point $P = (1/2, -1/2)$, whereas a triangle is a $\mathsf{cone}(Q, 1)$ where $Q = (1, 0)$. In the unit ℓ_1-disk, each sides of a square has length 1, as well as its diameter (its diagonals). A triangle has also diameter 1 (its hypotenuse), with both sides of length 1, and it forms an isosceles right triangle. Thus, each square region represents a fourth of the unit disk, possibly subdivided further into two equal triangles.

3 The makespan for L_1 is at most 5

At a high level, the proof consists of recruiting first a team of robots in a dense subregion, then these robots can wake up the other regions in parallel. The difficult part is to select these regions (triangles and squares) appropriately, depending on the number of sleeping robots and their distribution, and to prove that the bound holds in all the cases.

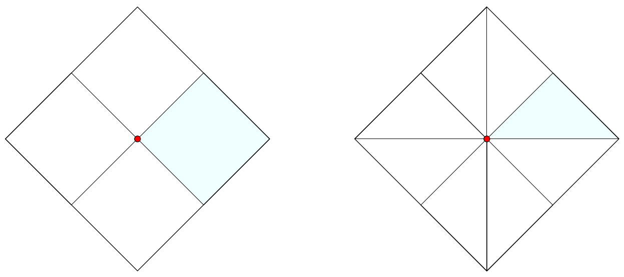

Figure 2 The unit ℓ_1-disk, divided into squares and triangles of diameter 1.

The proof relies on three key lemmas dealing with different wake-up processes in specific subregions, namely:

▶ **Lemma 3.** *A robot located at a corner of a square of diameter one can wake up any number $n \leq 5$ of robots in the square in two time units.*

Unfortunately, Lemma 3 cannot be extended beyond 5 robots. We can show that a makespan of 13/6 is required to wake up some configurations with 6 robots. It seems unavoidable to consider a case-based proof, which motivates the distinction between Lemma 3 and Lemma 4.

▶ **Lemma 4.** *A robot located at a corner of a square of diameter one can wake up 6 robots in the square and return to the origin with these robots in three time units.*

▶ **Lemma 5.** *A robot located at any of the three corners of a triangle T of diameter one, or two robots located at a same point on a side of T (not the hypotenuse) can wake up all the robots in T in two time units.*

A significant part of the paper is devoted to proving these lemmas. In particular, the proof of Lemma 5 is based on a complex recursive algorithm which is divided into 13 subcases.

Equipped with these lemmas, the proof of the main statement can be described as follows.

3.1 Proof of Theorem 1

It is based on the following strategy, called `L1-Strategy`, that is split into four scenarios as follows, depending on the number n_0 of robots in the densest square:

- $\boldsymbol{n_0 = 1}$. In this case, there are at most four robots to be awakened. The initiator wakes up one of them in one time unit. We now have two awake robots. Each of them independently wakes up another sleeping robot (if needed), in at most two time units (largest possible distance within the unit ℓ_1-disk). Then, any of the awake robots wakes up the last robot (if any) in at most two time units, which gives a total makespan of at most $1 + 2 + 2 = 5$.
- $\boldsymbol{2 \leq n_0 \leq 5}$. We recruit n_0 robots from the densest square S in two time units (Lemma 3), then come back to the origin (by time $2 + 1 = 3$) with $n_0 + 1 \geq 3$ awake robots. Since S is the densest square, three of the awake robots at the origin can each wake up one of the remaining squares (Lemma 3) in two time units, which gives a total of at most $3 + 2 = 5$.
- $\boldsymbol{6 \leq n_0 \leq 10}$. We recruit 6 robots (chosen arbitrarily) in the densest square S and move them to the origin in 3 time units (Lemma 4). Together with the initiator, this makes 7 robots. One of them wakes up the remaining robots in S, of which there are at most

4, in two time units (Lemma 3). The 6 others split into three teams of two robots, one team for each remaining square, and each robot wakes up half of the sleeping robots in its assigned square, again in two time units (Lemma 3), which gives a total of at most $3 + 2 = 5$.

- $\boldsymbol{n_0 \geq 11}$. The densest square S must contain a triangle T with at least $\lceil n_0/2 \rceil \geq 6$ sleeping robots. We wake up all the robots of T in 2 time units (Lemma 5) and move them to the origin. This makes at least 7 awaken robots at the origin. Each of them wakes up a remaining triangle in 2 time units (Lemma 5 again), which gives a total of at most $2 + 1 + 2 = 5$ time units.

This completes the upper bound of 5 on the makespan for L_1. Thanks to Theorem 2, a wake-up tree with such a makespan can be constructed in linear time, which completes the proof of Theorem 1.

3.2 Lemmas 3 and 4: monotonic paths

The full proofs of these lemmas are given in the Full Version [7]. We give here a summary of the main ideas. A path $(p_0, p_1, \ldots, p_t)$ is *monotonic* if it is both x-monotonic and y-monotonic. An essential feature of the ℓ_1-norm is that all monotonic paths are shortest paths. In other words, the length of a monotonic path equals the distance between its endpoints. A path is k-*monotonic* if it can be subdivided into at most k consecutive monotonic paths. A key remark is that within a region of diameter δ, the length of a k-monotonic path is at most $k\delta$.

For Lemma 3, we establish that for any set of at most 5 points, there exists a wake-up tree such that every branch is 2-monotonic.

For Lemma 4, a similar approach is taken by incorporating segments that return to the starting point and demonstrating that the resulting paths are 3-monotonic.

Observe that the monotonic nature of a path is solely determined by the relative arrangement of points in terms of their x and y coordinates. Therefore, our considerations are confined to a finite number of configurations (that can be indexed by permutations). See Figure 3.

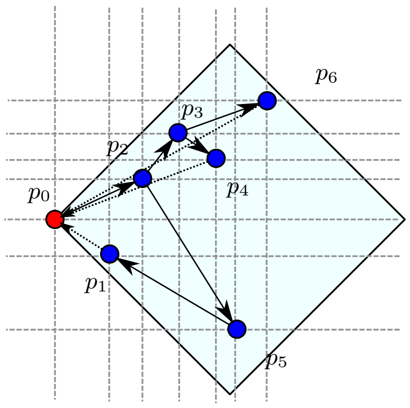

Figure 3 Illustration of a case of Lemma 4, the awake robot being at position p_0. Robots are first ordered w.r.t. their x-coordinate. p_0, p_2, p_3, p_6 is a 1-monotonic path whereas p_0, p_2, p_3, p_4 is a 2-monotonic path. p_0, p_2, p_5, p_1, p_0 is a 3-monotonic path. This wake-up tree is valid for any set of sleeping robots whose orders relative to the axes correspond to the permutation (3246517): the relative y-order of p_0 is 3, the one of p_1 is 2, ..., the one of p_6 is 7.

3.3 Lemma 5: recursive wake-up in triangles

Lemma 5 establishes that an arbitrary number of sleeping robots in a triangle T of diameter 1 can be woken up within two time units. The approach is inductive. Namely, waking up a triangle often reduces to waking up smaller nested triangles (containing strictly less sleeping robots), which explains why the statement of the lemma addresses several starting configurations. We present here a representative subset of three cases (out of thirteen). The other cases are presented in the Full Version [7].

▶ **Lemma 5.** *A robot located at any of the three corners of a triangle T of diameter one, or two robots located at a same point on a side of T (not the hypotenuse) can wake up all the robots in T in two time units.*

Because the unit ℓ_1-disk has a four-fold symmetry, we assume that the triangle T is oriented as in Figure 4, with vertices ABC and hypotenuse $[BC]$.

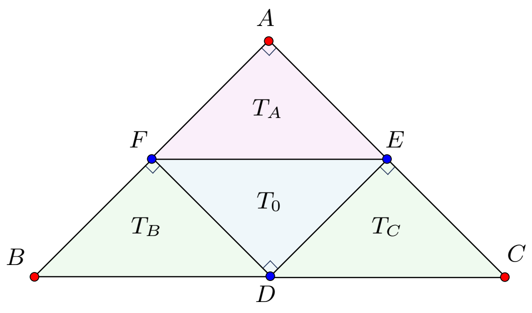

Figure 4 The triangle T with vertices $B = (0, 0)$, $C = (1, 0)$ and $A = (1/2, 1/2)$. Subdivision of the triangle T.

The goal is to show that all sleeping robots in T can be woken up in two time units, for each of the possible starting configurations. Up to symmetry, these configurations are: **Case A**: one awake robot is located in A; **Case B**: one awake robot is located in B; **Case C**: two awake robots are located at a same point along segment $[AB]$.

The strategy depends critically on how the robots are distributed within the triangle, which gives rise to a number of subcases (13 overall). Our proof is fully constructive (i.e., it yields an actual quadratic time algorithm that we have implemented) and we show in the Full Version [7] how to get a linear time algorithm. Technically, we proceed by induction on the number of sleeping robots in T. For cases A and B, at least one sleeping robot, at position p_1 in T, will be awake by the robot at $p_0 \in \{A, B\}$. For Case C, this is not necessarilly the case as an awake robot may simply move to another location without waking up any other robot. However, after one application of Case C, we check that Case C cannot immediately reapply. In other words, after two steps of induction, cases A or B will apply with one less sleeping robot.

The proof of Case B and C, and their subcases, relies on a regular subdivision of T into four smaller triangles of equal size. Call D, E, F the middle points of segments $[BC], [CA]$ and $[AB]$, respectively, and let T_A, T_B, T_C, T_0 be the triangles AFE, BDF, CED, and DEF (see Figure 4). Each of these triangles has diameter 1/2. Let $\varparallelogram P_B$ be the parallelogram $BDEF$. The diameter of $\varparallelogram P_B$ is 1.

We now present three of the thirteen cases (see Figure 5). These three cases are representative of the different arguments used. In these cases, the awake robot starts at point B, also referred to as p_0. The case analysis depends on the number of sleeping robots in $\varparallelogram P_B$. Namely, we apply Case B0 if it is empty, B1 if it contains one robot, and B2 if it contains two robots. A graphical summary of these three subcases is shown in Figure 5.

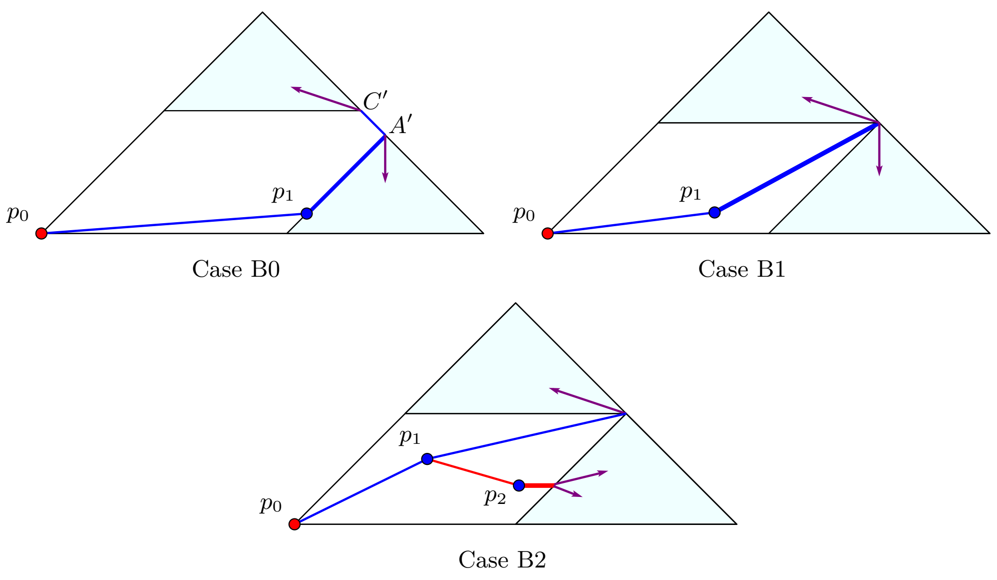

Figure 5 The subcases B0, B1 and B2 of Case B for Lemma 5. Regions in blue correspond to region where recursion occurs. Outgoing purple arrows indicates that the region will be woken up from the head location. A thick edge indicates that two awake robots follow the same path.

- **Case B0.** $\varparallelogram P_B$ is empty. We increase the size of $\varparallelogram P_B$ homothetically, keeping one of its corners at B, until a point p_1 is found (see Figure 5 - B0). The new parallelogram intersects with $[AC]$ in two points C' and A', where C' is the closest to A. This forms two smaller triangles which are homothetic to ABC. Because they result from intersecting a parallelogram, these triangles have the same size; namely, they have diameter $d = |AC'| = |A'C|$, which also implies that $|C'A'| = 1 - 2d$.
  The wake-up tree is as follows. The initial robot wakes up p_1. Depending on what side of the parallelogram p_1 lies on, both robots reach C' or A' using a path that is still monotonic from p_0, so they arrive before one time unit. One of them then reaches the other point (C' or A') in time $1-2d$. Finally, each robot wakes up one of the two triangles (separately) in time $2d$, recursing into case A and B (respectively). Overall, the makespan is thus $1 + (1 - 2d) + 2d = 2$.
- **Case B1.** $\varparallelogram P_B$ contains one robot. The robot at p_0 wakes up this robot, then both robots move to E before one time unit, since the path (p_0, p_1, E) is monotonic. Finally, one of them wakes up T_A (recursing in Case B) and the other T_C (recursing in Case A). These triangles have half the size of T, thus the makespan is at most $1 + 2 \cdot (1/2) = 2$.
- **Case B2.** $\varparallelogram P_B$ contains two robots at p_1 and p_2. W.l.o.g., assume $p_1 \leq_x p_2$. If (p_0, p_1, p_2) is monotonic, the strategy is the same as in Case B1: the robots reach point E in one time unit, then two of them wake up T_A and T_C independently. Otherwise, there exists a point C^* of $[DE]$ such that (p_1, p_2, C^*) is 1-monotonic. In this case, the initial robot wakes up the robot in p_1, then moves to E before one time unit and wakes up T_A in $2 \cdot (1/2) = 1$ time unit. Meanwhile, the robot in p_1 wakes up the robot in p_2 and both move to C^*.

  ▷ Claim 6. The 2-monotonic path (p_0, p_1, p_2, C^*) has length at most one.

  Proof. Let $p_1 = (x, y)$ and $C^* = (x', y')$. By 2-monotonicity, the length of the path is $|p_0p_1| + |p_1C^*| = (x + y) + ((x' - x) + (y - y')) = 2y + x' - y'$. In terms of y-coordinate, the height of T is $1/2$, thus the height of $\varparallelogram P_B$ is $1/4$, and $y \leq 1/4$. Moreover, because C^* lies on $[DE]$, we have $y' = x' - 1/2$, so $2y + x' - y' \leq 1/2 + x' - (x' - 1/2) = 1$. ◁

We thus have two robots located at C^* before one time unit. These robots can wake up T_C (of diameter $1/2$) in one time unit, by recursing in Case C.

4 Linear time algorithm

4.1 Heap-Strategy and linear time for L_1

A crude analysis of L1-Strategy shows that its complexity is quadratic. Even if we use a suitable data structure, we believe that the current proof cannot lead to a better than $\Omega(n \log n)$ time algorithm.

To obtain a linear algorithm, we will combine it with an algorithm that runs in linear time and achieves a makespan of $3 + O((\log n)/\sqrt{n}\,)$. Therefore, for small values of n (up to a characterized constant), we use L1-Strategy, and for larger values, we use the latter.

The strategy, called Generic-Strategy, is as follows: in a first phase, we divide the disk into $\sqrt{n}$ cones of arc-length $w = 2\pi(\ell_1)/\sqrt{n} = 8/\sqrt{n}$. Initially, we awake the robots in the densest cone using a strategy called Heap-Strategy. This particular cone contains at least $\sqrt{n}$ robots. In a second phase, we assign one awake robot to each of the remaining cones, in parallel, every ”sleeping” cone is awaken using Heap-Strategy again. Therefore, the makespan of the Generic-Strategy is no more than $1 + 2C(w, \sqrt{n}) = 1 + 2C(8/\sqrt{n}, \sqrt{n})$, where $C(w, n)$ is the wake-up time for n points in a cone of arc-length w.

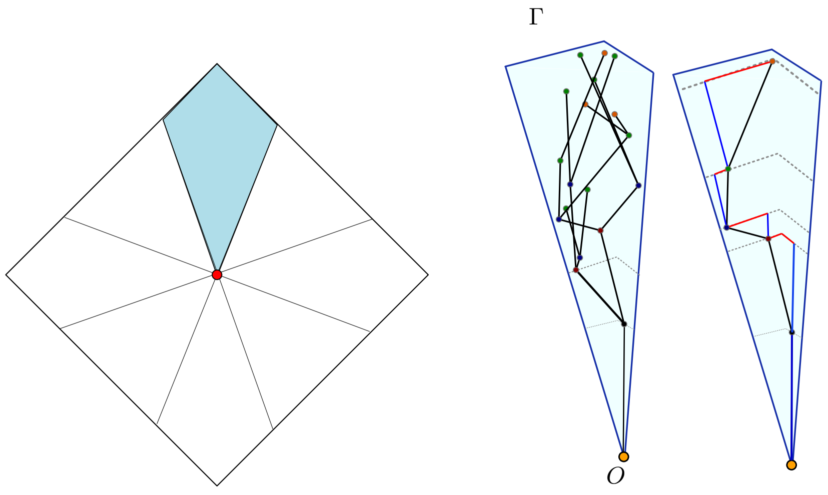

Figure 6 On the left, an arbitrary partition of the unit ℓ_1-disk into cones. In the middle, a wake-up tree computed using Heap-Strategy for an arbitrary cone (Γ represents an arc for an arbitrary norm). On the right, the analysis of the length of a branch drawn in black. In blue (resp. red) the radial (resp. angular) displacement of each edge.

To wake up a cone from p_0 containing points $p_1, p_2, \ldots, p_n$, we use the Heap-Strategy that simply constructs a minimum heap (binary) tree whose key is the ℓ_1-distance from the origin. By design, every path from p_0 to a point p_i has a *non-decreasing distance property* of the nodes w.r.t. the root.

More precisely, Heap-Strategy consists in building a minimum (binary) heap tree H for $\{p_1, \ldots, p_n\}$ where the key of p_i is the distance from p_0 to p_i. The wake-up tree rooted at p_0 is then composed of H itself, plus the edge connecting p_0 to the root of H (its top element), i.e., the closest point from p_0. Using the standard “build-heap” and “heapify” routines, H and thus the wake-up tree can be constructed in time $O(n)$.

Although we focus on L_1 in this section, the following proposition holds for any norm η:

▶ **Proposition 7.** *If P is contained in a cone of arc-length w, then* `Heap-Strategy` *constructs in time $O(n)$ a wake-up tree for P, rooted at the origin, with makespan at most $1 + w \lfloor \log_2 n \rfloor$.*

Proof. We can bound the length of an edge by the radial displacement plus the angular displacement. For each edge, the angular displacement is bounded by the arc-length w (see Figure 6). Furthermore, along a branch, the total radial displacement is bounded by the depth of the cone. Since the wake-up tree is a binary tree, each branch has at most $\lfloor \log_2 n \rfloor$ edges. Hence we obtain a makespan of $1 + w \lfloor \log_2 n \rfloor$ for the `Heap-Strategy` on a cone of arc-length w. ◀

Thus, from Proposition 7, we have $C(w, n) = 1 + w \lfloor \log_2 n \rfloor$, and the makespan of the `Generic-Strategy` for L_1 is smaller than $3 + 16 \log_2 n/\sqrt{n}$. So, for n large enough, that is as soon a $3 + 16 \log_2 n/\sqrt{n} \leq 5$, we can use this generic strategy to get a linear time for L_1, and we use the `L1-Strategy` otherwise, both with a guarantee of 5 on the makespan.

4.2 Linear time for arbitrary norms

We can generalize the generic strategy to arbitrary norms for any admissible bound on the makespan. The main difference is that the threshold $n_0 \leq n$ to use the generic algorithm depends on the wanted upper bound $\tau > 3$ assuming that $\tau \geq \gamma(\eta)$. For L_1, we know that $\gamma(\ell_1) = 5$ and we show a solution for $\tau = 5$.

Thus we get:

▶ **Theorem 2.** *Let η be any norm and let $\tau > 3$ be any real such that $\tau \geq \gamma(\eta)$. In time $O(n)$, a wake-up tree of makespan at most τ, rooted at the origin, can be built for any set of n points in the unit η-disk.*

Proof. Assume that $\tau > 3$ and $\tau \geq \gamma(\eta)$. We use the generic strategy presented for L_1. Since it is well-known that $\pi(\eta) \in [3, 4]$, we compute the least integer n_0 such that[2] $3 + 16 \log_2 n_0/\sqrt{n_0} \leq \tau$. Note that n_0 is a fixed constant, independent of n.

- If $n \geq n_0$, then we can apply `Generic-Strategy` providing a makespan that is less than τ.
- If $n < n_0$, then we use can brute-force algorithm for finding an optimal wake-up tree whose makespan is at most $\gamma(\eta)$ by definition of $\gamma(\eta)$. This is also at most τ by the choice of τ. The number of wake-up trees we have to consider in a brute-force algorithm is at most the number of labeled binary trees on n vertices that is $n! \cdot C_n = (2n)!/(n+1)! \leq (2n_0)!/(n_0+1)! = O(1)$ since n_0 is constant, where $C_n = \binom{2n}{n}/(n+1)$ is the nth Catalan's number. And, checking the makespan of each of these trees costs $O(n) = O(n_0) = O(1)$. ◀

5 Wake-up constants for other norms

Observe that `Heap-Strategy` suffers from having an unbounded makespan of $1 + w \log_2 n$ if $w = \omega(1/\log n)$. We address this as follows. First, we introduce a new strategy: `Split-Cone-Strategy` in order to remove this dependency to get a makespan of[3] $1 + \varphi w$ but

[2] For $\eta = \ell_1$, this integer appears to be $n_0 = 11\,665$. In the detailed proof of Theorem 2, in the Full Version [7], we show that we can use `Linear-Split-Strategy` instead of `Split-Cone-Strategy` in the `Generic-Strategy`, and that it is enough to choose the least n_0 such that $3 + 26/\sqrt{n_0} \leq \tau$. For $\eta = \ell_1$, this improved strategy gives an $n_0 = 169$.

[3] Recall that $\varphi = (1 + \sqrt{5})/2$ is the golden ratio.

running in time $O(n \log n)$. Then, we use `Split-Cone-Strategy` to get new bounds on the makespan for arbitrary norms. At the end of the section, we give some hints to get a linear time version of `Split-Cone-Strategy`, the full proof being presented in the Full Version [7].

Notations for arbitrary norms. In this paper, we concentrate our attention to the plane $\mathbb{R}^2$. Given a norm η, the *unit disk w.r.t.* η, or the unit η-disk for short, is the normed linear subspace of $(\mathbb{R}^2, \eta)$ induced by all the points at distance at most one from the origin, where distances are measured according to η, the distance between u and v being $\eta(v-u)$. The unit η-disk can be an arbitrary convex body that is symmetric about the origin. Note that the unit ℓ_2-disk[4] is a usual disk whereas the unit ℓ_1-disk is a square, rotated by 45 degrees with respect to the coordinate axes.

5.1 `Split-Cone-Strategy`

Let us describe the construction of the wake-up tree corresponding to `Split-Cone-Strategy`: (1) the initial awake robot located at this origin, wakes up the closest robot located at position p_1 w.r.t. the η-distance; (2) we split the current cone of arc-length w into two subcones of arc-length w/φ and $(1-1/\varphi)w$; (3) the current point p is linked to the closest point in the non-decreasing order in each of the two subcones; (4) each subcone is subdivided recursively according to Steps 2 and 3 until every point of the initial cone belongs to the binary wake-up tree.

It turns out that the subcone assigned to a point at depth i in the wake-up tree has an arc-length at most w/φ^i. After a precise analysis of the wake-up tree defined by `Split-Cone-Strategy`, we get:

▶ **Proposition 8.** *If P is contained in a cone of arc-length w, then* `Split-Cone-Strategy` *constructs in time $O(n \log n)$ a wake-up tree for P, rooted at the origin, of makespan at most $1 + \varphi w$.*

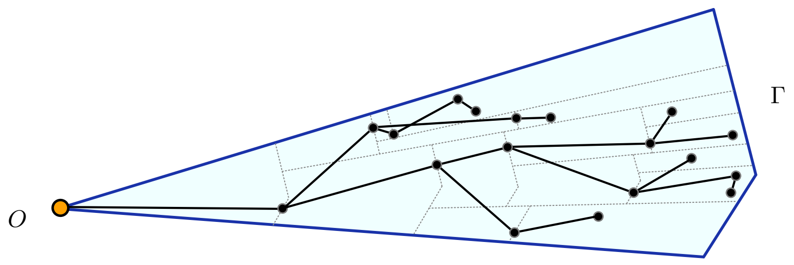

Figure 7 Illustration of the `Split-Cone-Strategy`, producing non-decreasing wake-up trees. The arc Γ is of length w in the η-norm. Note that edges can cross in such wake-up trees.

Let us sketch the proof of Proposition 8. To analyze the length of the longest branch of the wake-up tree (the makespan), we proceed as in the case of `Heap-Strategy`: we bound the length of each edge by its radial movement plus its lateral movement. The sum of the radial movements of a branch is at most the radius of the cone, that is 1. In addition, the lateral movement of the i-th edge is at most w/φ^{i-1}. So the sum of the lateral movements along a branch is at most φw. Hence the makespan of `Split-Cone-Strategy` is at most $1 + \varphi w$.

[4] For convenience, and to avoid extra notation, we use the same "unit η-disk" terminology to denote the normed subspace and, like here, its support, that is the set of all points/vectors of norm at most 1 (the unit disk).

If we combine `Split-Cone-Strategy` on the $O(n/\log n)$ closest points from the initial position and `Heap-Strategy` on the remaining points in tiny subcones, we have (see the Full Version [7] for the proofs):

▶ **Proposition 9.** *If P is contained in a cone of arc-length w, the* `Linear-Split-Strategy` *constructs in time $O(\lambda n)$ a wake-up tree for P, rooted at the origin, with makespan at most $1 + \varphi w + w \cdot (\log_2 n)^3/(\lambda n)$, for every $\lambda \geq 1$.*

5.2 Lower bounds

In L_1, consider four sleeping robots at positions $(\pm 1, 0)$ and $(0, \pm 1)$. Any wake-up tree spanning more than four points must have (unweighted) a depth at least 3. Then, the first hop has length 1, and the next two hops have length 2 (as the four points are mutually at distance 2), which overall gives a makespan of at least 5 for any wake-up tree. In Theorem 10, we give a generalization of this argument for any norm η, leading to an intriguing open question of matching this lower bound for other norms (see Conjecture 14).

Given a norm η, let us define $\Lambda(\eta)$ as half the perimeter of the largest inscribed parallelogram in the unit η-disk measured in the η-metric. (For the ℓ_1-norm, this perimeter corresponds to the circumference of the disk itself, and so $\Lambda(\ell_1) = 4$.) This is a classical parameter of 2D normed spaces. It can be formally defined by (see [23, 13]),

$$\Lambda(\eta) \;=\; \sup_{\substack{u,v \in \mathbb{R}^2 \\ \eta(u),\eta(v) \leq 1}} \left\{ \,\eta(u+v) + \eta(u-v)\, \right\} .$$

It is easy to check that $\Lambda(\eta) \in [2, 4]$. For general norms, the constant $\Lambda(\eta)$ can be difficult to calculate precisely. However, it is known (see [13, Proposition 1] for instance), that, for every $p \in [1, \infty]$, $\Lambda(\ell_p) = 2^{1+\max(1/p, 1-1/p)}$. In particular, $\Lambda(\ell_1) = \Lambda(\ell_\infty) = 4$ and $\Lambda(\ell_2) = 2\sqrt{2}$.

The wake-up constants for $n \in \{0, 1, 2, 3\}$ are easy to calculate. We have $\gamma_0(\eta) = 0$, $\gamma_1(\eta) = 1$, $\gamma_2(\eta) = \gamma_3(\eta) = 3$, and also $\gamma_n(\eta) \geq 3$ for all[5] $n \geq 3$. Our next result gives the exact value for $\gamma_4(\eta)$.

▶ **Theorem 10.** *For any norm η, $\gamma_4(\eta) = 1 + \Lambda(\eta)$.*

This implies a general lower bound of $\gamma(\eta) \geq 1 + \Lambda(\eta)$, for any norm η.

5.3 Upper bounds, ℓ_p-norms, and the conjecture

Wake-up cones of arc-length w with a makespan $1 + \varphi w$ allow us to state a first general upper bound: once two robots are awake at the origin (this can be done in at most two time units), each one can wake up half of the unit disk with arc-length $\pi(\eta)$. By this way, we can bound the wake-up constant for every norm:

▶ **Corollary 11.** *For any norm η, $\gamma(\eta) < 3 + \varphi\pi(\eta) \leq 9.473$.*

Note that since $\Lambda(\eta) \geq 2$, by letting $\tau = 1 + \Lambda(\eta)$, Theorem 2 simplifies and rewrites in the following meta-theorem:

▶ **Corollary 12.** *For any norm η with $\Lambda(\eta) > 2$, one can construct in time $O(n)$ a wake-up tree of makespan at most $\gamma(\eta)$ for any set of n points in the unit η-disk and rooted at the origin.*

[5] For $n \geq 2$, it is enough to place one sleeping robot at $(1, 0)$ and the $n - 1$ others at $(-1, 0)$.

Now, combining Theorem 1, Theorem 10 (with $\eta = \ell_p$), and standard inclusion arguments of unit ℓ_p-disk, we get the following bounds for the ℓ_p wake-up constant, which are better than what one can get by Corollary 11 with $\pi(\ell_p)$:

▶ **Corollary 13.** *For every $p \in [1, \infty]$, $1 + 2^{1+\max(1/p, 1-1/p)} \le \gamma(\ell_p) \le 5 \cdot 2^{\min(1/p, 1-1/p)}$.*

In the light of the lower bound $\gamma(\eta) \ge 1 + \Lambda(\eta)$ implied by Theorem 10, we propose the following natural conjecture.

▶ **Conjecture 14.** *For any norm η, $\gamma(\eta) = 1 + \Lambda(\eta)$.*

According to Theorem 10, which states that the bound $1 + \Lambda(\eta)$ is reached by $n = 4$ robots and doing some experiments, Conjecture 14 can be captured in the aphorism:

Wake up n robots is quicker than wake up four.

Theorem 1 and Corollary 13 prove the conjecture for $\eta \in \{\ell_1, \ell_\infty\}$. For $\eta = \ell_2$, if true, Conjecture 14 combined with Theorem 2 implies that in time $O(n)$ one can construct a wake-up tree of makespan $1 + 2\sqrt{2} \approx 3.82$. Using an analysis of the `Generic-Strategy` combined with the `Split-Cone-Strategy` (see the Full Version [7]), we can show that Conjecture 14 is true for ℓ_2 whenever $n \ge 551$.

6 Conclusion

In this article, we showed that a wake-up tree can be built in linear time with a makespan at most five for robots in L_1. This wake-up constant "five" is optimal: no strategy can guarantee less than five times the radius under the ℓ_1-norm. Our results imply a new upper bound of 7.07 for the ℓ_2-norm, improving upon the existing bound of 10.06, and so, in linear time. Some of our results are general enough to apply to every norm, implying an upper bound of 9.473 for every norm by introducing new algorithmic strategies, namely `Heap-Strategy` and `Split-Cone-Strategy`. We also showed how to get in linear time a wake-up tree of makespan no more than the wake-up constant, for every norm. The construction could be used in another setting since it provides a subcubic geometric graph with small diameter, namely at most $2\gamma(\eta) < 2 \cdot (3 + \varphi\pi(\eta))$ times the radius of the point set (Corollary 11).

Along the way, we conjecture that, for every norm η, the wake-up constant is $1 + \Lambda(\eta)$, where $\Lambda(\eta)$ is half the perimeter of the largest inscribed parallelogram in the unit disk of $(\mathbb{R}^2, \eta)$. According to our results, the conjecture is equivalent to saying that waking up n robots is always faster than waking up four robots. This conjecture is proved for the special cases of ℓ_1 and ℓ_∞ norms. As a first step towards proving the full conjecture, it would be interesting to determine the status of the ℓ_2-norm whose wake-up constant, according to our conjecture, should be $1 + 2\sqrt{2} \approx 3.82$. Among ℓ_p-norms, ℓ_2 is the norm whose current gap between the upper and lower bounds is the largest.

We also showed that the wake-up constant for fixed n asymptotically decreases with n, i.e., $\gamma_n(\ell_2) < \gamma_4(\ell_2)$ for large n (above 500), but we were unable to show that this inequality occurs for small n (say, 10). Surprisingly, some experiments that we conducted (see the Full Version [7]) show that at least one of the two following statements is wrong: (1) the wake-up constant is reached for points that are equally distributed along the boundary of the unit circle; and (2) for every $n \ge 4$, $\gamma_{n+2}(\ell_2) < \gamma_n(\ell_2)$.

In summary, the main open questions are the following:

- Is the wake-up constant of the ℓ_2-norm equal to $1 + 2\sqrt{2}$?
- Is the wake-up constant of the regular-hexagonal-norm[6] equal to 4?
- Is Conjecture 14 true for all ℓ_p-norms? And if so, is it true every norm?
- Does a linear time PTAS exists?
- What is the status of higher dimensions?

References

1 Zachary Abel, Hugo A. Akitaya, and Yu Jingjin. Freeze tag awakening in 2D is NP-hard. In *27th Annual Fall Workshop on Computational Geometry (FWCG)*, November 2017. URL: `https://www.ams.stonybrook.edu/~jsbm/fwcg17/proceedings.html`.

2 Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, Joseph S.B. Mitchell, and Martin Skutella. The freeze-tag problem: How to wake up a swarm of robots. In *13th Symposium on Discrete Algorithms (SODA)*, pages 568–577. ACM-SIAM, 2002. URL: `http://dl.acm.org/citation.cfm?id=545381.545457`.

3 Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, Joseph S.B. Mitchell, and Martin Skutella. The freeze-tag problem: How to wake up a swarm of robots. *Algorithmica*, 46:193–221, 2006. `doi:10.1007/s00453-006-1206-1`.

4 Esther M. Arkin, Michael A. Bender, Dongdong Ge, Simai He, and Joseph S.B. Mitchell. Improved approximation algorithms for the freeze-tag problem. In *15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 295–303. ACM Press, June 2003. `doi:10.1145/777412.777465`.

5 Amitai Armona, Adi Avidora, and Oded Schwartz. Cooperative tsp. *Theoretical Computer Science*, 411(31-33):2847–2863, June 2010. `doi:10.1016/j.tcs.2010.04.016`.

6 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, September 1998. `doi:10.1145/290179.290180`.

7 Nicolas Bonichon, Arnaud Casteigts, Cyril Gavoille, and Nicolas Hanusse. Freeze-tag in L1 has wake-up time five. Technical Report 2402.03258v1 [cs.DS], arXiv, February 2024. `doi:2402.03258v1`.

8 Josh Brunner and Julian Wellman. An optimal algorithm for online freeze-tag. In *10th International Conference Fun with Algorithms (FUN)*, volume 157 of *LIPIcs*, pages 8:1–11, September 2020. `doi:10.4230/LIPIcs.FUN.2021.8`.

9 Dan George Bucatanschi. The ant colony system for the freeze-tag problem. In *Midstates Conference on Undergraduate Research in Mathematics and Computer Science (MCURCSM)*, pages 61–69, 2004.

10 Dan Georges Bucatanschi, Blaine Hoffmann, Kevin R. Hutson, and R. Matthew Kretchmar. A neighborhood search technique for the freeze tag problem. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, volume 37 of *Operations Research/-Computer Science Interfaces Series*, pages 97–113, 2007. `doi:10.1007/978-0-387-48793-9_7`.

11 Shantanu Das. Graph explorations with mobile agents. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities*, volume 11340 of *Lecture Notes in Computer Science*, chapter 16, pages 403–422. Springer, Cham, 2019. `doi:10.1007/978-3-030-11072-7_16`.

12 L. Few. The shortest path and the shortest road through n points. *Mathematika*, 2(2):141–144, 1955. `doi:10.1112/S0025579300000784`.

[6] With this norm, it is easy to show that its unit disk (a regular hexagon) contains inscribed parallelograms of half perimeter 3. This is clearly the largest possible length since 3 is also the half-perimeter of this disk (a hexagon).

13 Jil Gao. Normal structure and the arc length in banach spaces. *Taiwanese Journal of Mathematics*, 5(2):353–366, June 2001. URL: `http://www.jstor.org/stable/43828249`.

14 Mikael Hammar, Bengt J. Nilsson, and Mia Persson. The online freeze-tag problem. In *7th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 3887 of *Lecture Notes in Computer Science*, pages 569–579. Springer, March 2006. `doi:10.1007/11682462_53`.

15 Matthew Johnson. Easier hardness for 3D freeze-tag. In *27th Annual Fall Workshop on Computational Geometry (FWCG)*, November 2017. URL: `https://www.ams.stonybrook.edu/~jsbm/fwcg17/proceedings.html`.

16 Marek Karpinski. Towards better inapproximability bounds for TSP: A challenge of global dependencies. In *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-097, June 2015. URL: `https://eccc.weizmann.ac.il/report/2015/097/`.

17 Hamidreza Keshavarz. Applying tabu search to the freeze-tag. In *1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 37–41. IEEE Computer Society Press, March 2016. `doi:10.1109/CSIEC.2016.7482136`.

18 Jochen Könemann, Asaf Levin, and Amitabh Sinha. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2):117–129, 2005. `doi:10.1007/s00453-004-1121-2`.

19 Joseph S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. `doi:10.1137/S0097539796309764`.

20 Zahra Moezkarimi and Alireza Bagheri. A PTAS for geometric 2-FTP. *Information Processing Letters*, 114(12):670–675, 2014. `doi:10.1016/j.ipl.2014.06.017`.

21 Lehilton Lelis Chaves Pedrosa and Lucas de Oliveira Silva. Freeze-tag is NP-hard in 3D with L_1 distance. In *12th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS)*, volume 223:C, pages 360–366. Procedia Computer Science, September 2023. `doi:10.1016/j.procs.2023.08.248`.

22 Carsten Rössner and Jean-Pierre Seifert. Hardness of approximating shortest integer relations among rational numbers. *Theoretical Computer Science*, 209(1-2):287–297, December 1998. `doi:10.1016/S0304-3975(97)00118-7`.

23 Juan Jorge Schäffer. *Geometry of Spheres in Normed Spaces*, volume 20 of *Lecture Notes in Pure and Applied Mathematics*. Dekker, Marcel, 1976.

24 Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *24th AAAI Conference on Artificial Intelligence (AAAI)*, volume 24, pages 173–178. AAAI Press, July 2010. `doi:10.1609/aaai.v24i1.7564`.

25 Marcelo O. Sztainberg, Esther M. Arkin, Michael A. Bender, and Joseph S.B. Mitchell. Theoretical and experimental analysis of heuristics for the "freeze-tag" robot awakening problem. *IEEE Transactions on Robotics*, 20(4):691–701, August 2004. `doi:10.1109/TRO.2004.829439`.

26 Vera Traub, Jens Vygen, and Rico Zenklusen. Reducing path TSP to TSP. In *52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 14–27. ACM Press, June 2020. `doi:10.1145/3357713.3384256`.

27 Ehsan Najafi Yazdia, Alireza Bagheri, Zahra Moezkarimia, and Hamidreza Keshavarz. An $O(1)$-approximation algorithm for the 2-dimensional geometric freeze-tag problem. *Information Processing Letters*, 115(6-8):618–622, June 2015. `doi:10.1016/j.ipl.2015.02.011`.