Distributed Freeze Tag: a Sustainable Solution to Discover and Wake-up a Robot Swarm

Cyril Gavoille
LaBRI, Université de Bordeaux, CNRS
Bordeaux, France
gavoille@labri.fr

Nicolas Hanusse
LaBRI, Université de Bordeaux, CNRS
Bordeaux, France
hanusse@labri.fr

Gabriel Le Bouder
LaBRI, Université de Bordeaux, CNRS
Bordeaux, France
gabriel.le-bouder@labri.fr

Taïssir Marcé
LaBRI, Université de Bordeaux, CNRS
Bordeaux, France
taissir.marce@labri.fr

Abstract

The Freeze-Tag Problem consists in waking up a swarm of robots starting with one initially awake robot. While there exists a wide literature on the centralized setting, where the locations of the robots are known in advance, we focus on the distributed version where the locations of the robots, $\mathcal{P}$, are unknown, and where awake robots only detect other robots up to distance 1. Assuming that moving at distance δ takes a time δ, we show that waking up the whole swarm takes $O(\rho + \ell^2 \log(\rho/\ell))$, where ρ is the largest distance from the initial robot to any point of $\mathcal{P}$, and ℓ is the connectivity threshold of $\mathcal{P}$. Moreover, the result is complemented by a matching lower bound. We also provide other distributed algorithms, complemented with lower bounds, whenever each robot has a bounded amount of energy.

CCS Concepts

• **Theory of computation → Distributed algorithms**; **Computational geometry**; *Distributed computing models.*

Keywords

Mobile Agents, Distributed Algorithms, Collaborative Exploration

ACM Reference Format:
Cyril Gavoille, Nicolas Hanusse, Gabriel Le Bouder, and Taïssir Marcé. 2025. Distributed Freeze Tag: a Sustainable Solution to Discover and Wake-up a Robot Swarm. In *ACM Symposium on Principles of Distributed Computing (PODC '25), June 16–20, 2025, Huatulco, Mexico.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3732772.3733538

1 Introduction

In order to save energy in distributed systems, the paradigms of sleeping models and algorithms have recently received attention. Nodes or robots are, by default, inactive or on standby: the energy consumption is negligible and these periods can be used to harvest energy. A robot becomes active only when it is required.

The Freeze-Tag Problem (FTP) consists in waking up a swarm of n inactive (or sleeping) robots as fast as possible, assuming that one robot is initially active, the source s. To be woken up, a sleeping robot has to be reached by one of the awake robots, which can move in the plane. Once a robot becomes active, it can help wake up other robots.

The FTP has been introduced in a centralized setting, where the n locations of the sleeping robots are known by the initially awake robot s. In this article, we propose a distributed version of the FTP: (1) the locations of the sleeping robots are not known in advance; (2) using a local snapshot, active robots only have a distance-1 visibility; and (3) robots need to be co-located to communicate. Note that due to the visibility constraint, it may be required to explore further than radius 1 to locate sleeping robots.

In order to get the most sustainable solution in a long-life perspective, we aim at minimizing the energy consumption. In particular, since moving is identified as an energy intensive task, the goal is to minimize *the makespan*, that is the time to wake up every robot, assuming unitary speed of the robots, i.e., moving at distance δ takes δ unit of time. It is also assumed that robots use discrete snapshots only, since continuous snapshots may not be energy friendly.

1.1 State of the Art

Any solution to the FTP can be seen as a rooted tree spanning the set of the $n+1$ robots' positions, called the *wake-up tree* [3]. The root corresponds to the position of s, has one child, and each of the n other nodes has at most two children. An edge from node p to its child p' is weighted by the distance between p and p', and corresponds to the path taken by a robot positioned at p to awake a robot at p'. The makespan of a solution corresponds to the (weighted) depth of the wake-up tree. In particular, a solution with optimal makespan has a wake-up tree with minimum depth.

Freeze Tag. Even in simple cases, an optimal solution to the FTP cannot be computed in polynomial time. Arkin *et al.* [3] showed that, even on star metrics, FTP is NP-hard. Moreover, they proved that getting an 5/3-approximation is NP-hard for general metrics on weighted graphs. However, in [4], the authors give a polynomial time algorithm to compute an $O(1)$-approximation for general graphs, assuming one sleeping robot per node.

PODC '25, Huatulco, Mexico

ACM ISBN 979-8-4007-1885-4/25/06
https://doi.org/10.1145/3732772.3733538

In this paper, we focus on the *geometric* setting, where robot movements have no restrictions and where the position set $\mathcal{P}$ lies on the Euclidean plane. Even in this setting, the problem remains NP-hard [1]. It has been shown by Yazdi *et al.* [22] that a wake-up tree of makespan of at most 10.07ρ can be (sequentially) computed in time $O(n)$, where ρ is the largest distance from s to any point of $\mathcal{P}$. The constant 10.1, aka the *wake-up constant* of the Euclidean plane, was later improved by Bonichon *et al.* [5] to 7.07. More generally, they proved that the wake-up constant for *any* norm is no more than 9.48, and that a corresponding wake-up tree can be computed in time $O(n)$. Very recently, the upper bound dropped independently to 5.06 [2] and to 4.63 [6]. It is known that $1+2\sqrt{2} \approx 3.83$ is a lower bound on the wake-up constant of the plane [5].

A first step toward the computation of a wake-up tree without a global knowledge of the robot's positions is the *online setting* [8, 20]. In this case, each robot only appears at a specified time that is not known in advance. In [8], the authors propose a solution with a competitive ratio of $1 + \sqrt{2}$ w.r.t. to the optimal partial wake-up tree.

Collaborative Exploration. Obviously, any collaborative exploration problem requires a team of active robots. Conversely, the *distributed FTP* (dFTP for short), requires exploring some area to discover sleeping robots and thus is naturally connected to exploration of the plane with one or more robots. The survey of Das [11] contains many references to such exploring problems in unknown graphs. In the dFTP, we start with one active robot, and after few steps, we can have k active robots. So, the task of discovering new robots can indeed be seen as a collaborative exploration task. The question of improving exploration with the use of $k > 1$ robots is challenging and widely open. For instance, it has been shown by Fraigniaud *et al.* [17] that in unweighted trees of diameter D, distributed exploration can be done in $O(D + n/\log k)$ unitary moves, even if robots are allowed to share some information through Read/Writes operations on nodes, whereas we could hope for a speed-up of k with $O(D + n/k)$ unitary moves. However, if the underlying graph is a two dimensional sub-grid of n vertices, a grid with rectangular holes, Ortlof and Schindelhauer [21] show how to get an optimal speedup of factor k.

Discovering a robot at distance D with k co-located robots in the plane can be done within $\Theta(D + D^2/k)$ unitary moves per robot using either parallel spiral trajectories [18], or by partitioning a square of width D into k rectangles of width D/k and height D. This problem, aka Treasure Hunt Problem or Cow-Path Problem, has been widely studied for $k = 1$ or with imprecise geometry [7]. Interestingly, the authors of [15] have shown that the knowledge of an approximation of k is required to get the bound $\Theta(D + D^2/k)$. The question of the knowledge of k arises whenever the k robots do not start the exploration together (as in the dFTP), or whenever the communication ability of the robots is limited.

Energy Consumption. Some recent works deal with the problem of distributed tasks with energy constraints, or aim at minimizing the energy consumption. In the sleeping model [10], nodes or robots are either sleeping or active. If a robot is sleeping, its consumption is assumed to be negligible. The state of each robot in synchronized rounds is given by a centralized schedule. Distributed tasks such as coloring or MIS computations have been considered in the sleeping model.

Energy Constrained Exploration Problems are perhaps more related to our problem. For instance, in the *Piece-Meal Graph Exploration*, robots have a given budget for the energy and need to refuel at a home base before exploring unknown parts of the graph. Note that a solution of the treasure hunt in a grid graph can be used in the plane. In [13], the authors show that n-node and m-edge unweighted graphs of radius R can be explored by $k = 1$ agent with an energy budget $B = (1 + \alpha) \cdot R$ in $O(m + n/\alpha)$ unitary moves. For $k > 1$, [12, 14] deal with the Energy Constrained Depth First Search while minimizing the number k of robots requiered, each with an energy budget $O(R)$, to explore a tree of radius R. Other distributed algorithms for energy constrained agents have been considered in [9]. The authors show how to provide a feasible movement schedule for mobile agents for the Delivery Problem in which each agent has limited energy, which constrains the distance it can travel. Hence, multiple agents need to collaborate to move and deliver the package, each agent handing over the package to the next agent to carry it forward. However, the positions of the agents are assumed to be known and the computation is centralized.

However, all these results related to collaborative exploration and energy consumption are not directly related to our setting, essentially because we must face the fact that, by definition of the problem, the number of active robots collaborating keeps on increasing, from 1 to $n + 1$.

1.2 Model

Computational Model. We consider a swarm of robots in the Euclidean plane. Robots are all initially asleep, except one which we call the *source* and denote s, initially located at position $p_0 = (0, 0)$. The set of all robots is denoted by $\mathcal{R} = \{s, r_1, \ldots, r_n\}$, robots r_i being the initially asleep robots. We denote by p_i the initial position of robot r_i and by $\mathcal{P}$ the set of all initial positions of initially asleep robots, i.e., $\mathcal{P} = \{p_1, \ldots, p_n\}$. Robots are endowed with a visible light indicating their status (sleeping or awake), which can be observed by any active robot close enough (in its distance-1 vicinity). Sleeping robots are computationally inactive. They can neither move, observe, nor do any type of computation. Awake robots are aware of the absolute coordinate system, a same global clock and are able to locate and distinguish sleeping and awake robots in their vicinity, by using a function look. They can also share variables of their memory with co-located robots, and can operate computations based on the information they gathered previously. Finally, they can move in the plane, based on the computation they performed. Robots move at speed 1, which means it takes a time δ for a robot to move between any two points at Euclidean distance δ. Thus the behaviour of a robot can be described in the standard Look-Compute-Move Model (see [16]).For synchronization purposes, robots can also wait for any duration of time at a fixed position. When an awake robot and a sleeping robot are co-located, the awake one can wake the other one up, and possibly share with it some information as previously said.

An algorithm $\mathcal{A}$ aiming to solve the dFTP is executed in parallel by all the awake robots. The execution of $\mathcal{A}$ terminates when all active robots have terminated their computation and moves. The

execution is valid if, when it terminates, all the initially asleep robots have been awakened. The makespan of an execution is the duration between the beginning of the algorithm and its termination, which basically counts the duration of the moving and waiting actions of robots.

Robots have a local unlimited memory. Typically, they can store the positions of some robots and their status (sleeping/awake) at the time they see them in their vicinity. Note that robots can give themselves a globally unique identifier as soon as they are awakened, by storing their initial position. We will also consider the variant of the model where the robots are not free to move as long as they want, but are rather limited by some *energy budget* B. In this variant, a robot can move for a total distance at most B.

Spread of $\mathcal{P}$. Our results are closely tied to the distribution of $\mathcal{P}$. Typically, if the distance between every pair of robots is much larger than 1, a robot may incorrectly believe that it is alone[1] which significantly complicates the problem. To formally present our results as in Table 1, let us introduce some parameters related to $\mathcal{P} \subset \mathbb{R}^2$.

Given a real $\delta \geqslant 0$, and $\mathcal{X} \subset \mathbb{R}^2$, the *$\delta$-disk graph* of $\mathcal{X}$ is the edge-weighted geometric graph with vertex set $\mathcal{X}$, two points $u, v \in \mathcal{X}$ being connected by an edge if and only if u and v are at (Euclidean) distance at most δ, and the weight of the edge corresponds to the distance between their endpoints.

Let $(\mathcal{P}, s)$ be an n-point set $\mathcal{P} \subset \mathbb{R}^2$ with a source $s \notin \mathcal{P}$. The *radius* of $(\mathcal{P}, s)$, denoted by ρ^*, is the largest distance from s to any point of $\mathcal{P}$. The *connectivity-threshold* of $(\mathcal{P}, s)$, denoted by ℓ^*, is the smallest radius δ such that the δ-disk graph of $\mathcal{P} \cup \{s\}$ is connected. Given $\ell > 0$, the *ℓ-eccentricity* of $(\mathcal{P}, s)$, denoted by ξ_ℓ, is the – finite or infinite – minimum weighted-depth of a spanning tree of the ℓ-disk graph of $\mathcal{P} \cup \{s\}$ rooted at s.

Problem Definition. We shall suppose that s starts with some information about the connectivity-threshold, the radius, and the number of sleeping robots in $\mathcal{P}$ that it is supposed to wake up. Formally, a tuple of values (ℓ, ρ, n) is given to s at its start. Indeed, without any information, it is impossible to distinguish (for example) the case where $n = 0$ (s is alone) from the case $n > 0$ without moving for eternity, and therefore without termination. A dFTP algorithm with input (ℓ, ρ, n) should terminate on any n-point set $(\mathcal{P}, s)$ such that $\ell^* \leqslant \ell$ and $\rho^* \leqslant \rho$. Note that if $\rho \leqslant 1$ then every robot is seen by the source s and can be woken up in time $O(1)$ with energy budget $O(1)$ by solving the centralized version in s, e.g., as done in [5]. On the other hand, if $\ell \leqslant 1$ then the discovery of neighbors around robots becomes trivial. Furthermore, as stated in Lemma 2.6, every $(\mathcal{P}, s)$ satisfies $\ell^* \leqslant \rho^* \leqslant n\ell^*$. An input (ℓ, ρ, n) is said *admissible* if $\ell \leqslant \rho \leqslant n\ell$. Note that if (ℓ, ρ, n) is admissible, then $(\lceil \ell \rceil, \lceil \rho \rceil, n)$ is too. For the sake of simplicity, we suppose in the following that parameters ℓ and ρ are positive integers. The dFTP is formally defined as follows:

Definition 1.1 (dFTP). A distributed algorithm $\mathcal{A}$ solves the dFTP if, for any admissible tuple (ℓ, ρ, n), and for any n-point set $\mathcal{P}$ with source s such that $\rho^* \leqslant \rho$ and $\ell^* \leqslant \ell$, the execution of $\mathcal{A}$ on $\mathcal{P}$ at source s, given (ℓ, ρ, n), eventually wakes up all the robots and terminates. Moreover, it solves the dFTP with energy budget B if the previous holds, assuming that the total movement lengths of each robot does not exceed B.

[1] Except for the co-located robot that activated it.

1.3 Contributions

Our contributions are summarized in Table 1. We present three algorithms, called $\mathcal{A}_{Separator}$, $\mathcal{A}_{Grid}$ and $\mathcal{A}_{Wave}$. The first algorithm $\mathcal{A}_{Separator}$ solves the dFTP, with no limits on the energy budget, and has makespan $O(\rho + \ell^2 \log(\rho/\ell))$. This result is complemented by a matching lower bound.

The two other algorithms consider the dFTP with energy budget B. We first show that no algorithm can solve the limited energy budget variant if $B < c\ell^2$, for some constant $c > 0$. Then, for $B \in \Theta(\ell^2)$, i.e., as little energy as possible to solve the dFTP, $\mathcal{A}_{Grid}$ achieves a makespan of $O(\xi_\ell \cdot \ell)$. Using slightly more energy, namely $B \in \Theta(\ell^2 \log \ell)$, $\mathcal{A}_{Wave}$ has a significantly lower makespan, which matches a second lower bound we introduce.

Roadmap. In Section 2 we present the main building blocks of our algorithm. For the sake of readability, one consequent block is detailed in Section 5. These high-level procedures are used to present $\mathcal{A}_{Separator}$ in Section 3. In Section 4, we present two algorithms for the constrained energy setting, $\mathcal{A}_{Grid}$, and $\mathcal{A}_{Wave}$ for which both $\mathcal{A}_{Separator}$ and $\mathcal{A}_{Grid}$ are building blocks. Due to space limitation, we only provide the proofs of the main results, a complete version of this work is given in [19].

2 Building Blocks

Our algorithms are based on different routines and geometric notions such as exploring regions, computing and realizing wake-up trees, organizing teams of robots to explore regions in parallel, sampling of point sets in regions, and specific geometric separators.

2.1 Exploration (Explore)

One central task robots are led to realize is the exploration of a given region, in order to collect the positions of all the sleeping robots in that region. For the sake of simplicity, we only consider rectangular regions, whose orientation is parallel to the axis.

Lemma 2.1 (Explore). *There exists a procedure Explore such that, for any rectangle $\mathcal{S}$ of dimensions $w \times h$, for any two positions $p, p' \in \mathcal{S}$, the execution of Explore$(\mathcal{S}, p')$ at time t, by a team of k robots $\mathcal{T} = \{r_1, \ldots, r_k\}$ initially co-located at position p guarantees:*

- *it terminates at time t' with $(t' - t) \in O(wh/k + w + h)$; and*
- *at time t', robots of $\mathcal{T}$ terminates at p' and have gathered the initial positions of all robots of $\mathcal{S}$ that are asleep at t'.*

2.2 Realization of a Central Wake-up-Tree

In [5, 22] the authors show that, knowing the initial positions of robots, it is possible to compute a wake-up tree in linear time, whose makespan is an approximation of the optimal. Yet, in the distributed setting, some specific problems may arise. Indeed, two awake robots r_i and r_j may compute independently two wake-up trees on different but not disjoint subsets X_i and X_j of $\mathcal{P}$. If p_k, the position of r_k, belongs to $X_i \cap X_j$, then r_i and r_j are said *in conflict*. Both need to use r_k for their wake-up trees and only the first robot to reach r_k is able to do it (since r_k will then move). The second one only find out that r_k has left its initial position p_k when

Energy	Algorithm	Makespan	Lower Bound
unconstrained	$\mathcal{A}_{Separator}$	$O(\rho + \ell^2 \log(\rho/\ell))$ - Th. 3.1	$\Omega(\rho + \ell^2 \log(\rho/\ell))$ - Th. 3.6
$< \pi(\ell^2 - 1)/2$	-	-	unfeasible - Th. 4.1
$\Theta(\ell^2)$	$\mathcal{A}_{Grid}$	$O(\xi_\ell \cdot \ell)$ - Th. 4.2	$\Omega(\xi_\ell + \ell^2 \log(\xi_\ell/\ell))$ - Th. 4.4
$\Theta(\ell^2 \log \ell)$	$\mathcal{A}_{Wave}$	$O(\xi_\ell + \ell^2 \log(\xi_\ell/\ell))$ - Th. 4.3	

Table 1: Complexity of the makespan for the dFTP given (ℓ, ρ, n).

itself or one of its descendent arrives close to p_k. In this situation the consistency of the wake-up tree is broken, and a new wake-up tree has to be computed for the sub-tree rooted at r_k. Since this situation can be repeated an arbitrarily large number of times for a set $Y = X_i \cap X_j$, there is no evidence that r_j can wake up $X_j \setminus Y$ in a time proportional to the diameter of $X_j \setminus Y$. To deal with that issue, we ensure in our algorithms that wake-up trees are computed in separate regions of the plane, and that at most one robot computes a wake-up tree in a given region. This is formalized in Lemma 2.2.

Lemma 2.2 (Distributed Makespan). *Given a square region $\mathcal{S}$ of width R and a robot r positioned in the center of $\mathcal{S}$, knowing both $\mathcal{S}$ and a set of sleeping robots $\mathcal{R}_S$ whose initial positions are in $\mathcal{S}$, if no robots other than $\{r\} \cup \mathcal{R}_S$ takes action in $\mathcal{S}$, then r can wake-up all robots of $\mathcal{R}_S$ in time $5R$.*

More generally, if robots do not know the positions of sleeping robots in $\mathcal{S}$, it is possible for a robot to discover them before computing a wake-up tree. This process will be used in $\mathcal{A}_{Grid}$, and is presented in the following Corollary.

Corollary 2.3 (Explore and Wake up). *Given a square region $\mathcal{S}$ of width R containing at least one awake robot, if no awake robot goes through the border $\mathcal{S}$, it is possible to wake-up all sleeping robots of $\mathcal{S}$ in time $R^2 + (10 + \sqrt{2})R$.*

2.3 Geometric Separators

One central tool we use is *geometric separators*. Given $\ell \geqslant \ell^*$ and a square $\mathcal{S}$ of width R centered at position p, the interior of $\mathcal{S}$, including $\mathcal{S}$, is noted $\mathcal{S}^{in}$ and the remaining region of the plane is noted $\mathcal{S}^{out}$. We define the *separator of* $\mathcal{S}$, denoted $\text{sep}(\mathcal{S})$, as the region bounded by $\mathcal{S}$ on the one hand, and a square of center p and width $\max(0, R - 2\ell)$, on the other hand. We immediately get:

Lemma 2.4 (Separators). *Let $\mathcal{S}$ be a square, and consider $\ell \geqslant \ell^*$. Any path in the ℓ-disk graph of $\mathcal{P}$ connecting robots $r \in \mathcal{S}^{in}$ and $r' \in \mathcal{S}^{out}$ contains at least one robot located in* $\text{sep}(\mathcal{S})$.

In particular, if $\mathcal{P} \cap \text{sep}(\mathcal{S}) = \emptyset$, either $\mathcal{P} \subset \mathcal{S}^{in}$ or $\mathcal{P} \subset \mathcal{S}^{out}$.

2.4 Distributed ℓ-Sampling

To begin, we point out that the time to wake up every robot in a square region of width R, $O(R)$ (Lemma 2.2), is often dominated by the time to discover sleeping robots in this region by a team of k robots, which is $O(R^2/k)$ (Lemma 2.1). However, knowing an upper bound ℓ on the connectivity threshold can help, thanks to two independent notions: *ℓ-coverings* and *ℓ-samplings*. We say that $\mathcal{S}$ is *covered* by a set of positions $\mathcal{P}'$ if any robot within $\mathcal{S}$ is at distance at most ℓ from a robot whose position is in $\mathcal{P}'$. Given an set $\mathcal{P}'$ covering $\mathcal{P}$, such that robots positioned in $\mathcal{P}'$ are awake, it is possible to discover all the remaining sleeping robots in parallel, by having each robot exploring the ℓ-disk around its initial position. This takes time $O(R + \ell^2)$ if we do not suppose that robots are located at their initial position. After that, all robots can gather at the source s in time R and wake-up every robot in time $O(R)$ using a centralized wake-up algorithm (Lemma 2.2). Yet, an ℓ-covering set can be arbitrary large, and therefore may be excessively long to compute.

For that reason, we won't consider *any* ℓ-covering set, but one which is also an ℓ-sampling: *an ℓ-sampling of a region $\mathcal{S}$ is a subset of positions $\mathcal{P}' \subseteq \mathcal{P} \cap \mathcal{S}$ that are pairwise at distance at least ℓ*. The two main questions are now: how large can an ℓ-sampling be, and how to efficiently compute one in a distributive way. Lemma 2.5 answers the first question. We then present algorithm DFSampling, which solves the second question, as proven in Lemma 5.1.

Lemma 2.5 (ℓ-sampling cardinality). *If $\mathcal{P}'$ is an ℓ-sampling of a square region of width R, then $|\mathcal{P}'| \leqslant 16R^2/(\pi\ell^2)$.*

DFSampling is a distributed algorithm computing an ℓ-sampling of a squared region $\mathcal{S}$. A single robot or a co-located team of robots starts from a position of $\mathcal{P} \cap \mathcal{S}$ and aims at finding an ℓ-sampling of size 4ℓ (See Figure 1b). This sampling represents positions of robots to be later recruited to become a new team. This sampling is discovered using a specific exploration algorithm. Since it may happen that exploring from one single position is not enough to reach a sample of size 4ℓ, the team may use several starting positions for the exploration task, which we call the *seeds* $\mathcal{X} \subset \mathcal{S}$. In Figure 1c, $\mathcal{X}$ is the set of points within an ℓ-separator of a sub-square.

Roughly speaking, DFSampling is based on a Depth-First Search in the 2ℓ-disk graph of $\mathcal{P} \cap \mathcal{S}$, starting by the position of seeds from $\mathcal{X}$. Whenever a point p is discovered, it is added to $\mathcal{P}'$ only if it is at distance greater than ℓ from any other points already added to $\mathcal{P}'$. This is required to guarantee that $\mathcal{P}'$ is indeed a ℓ-sampling. A detailed description is given in Section 5.

2.5 Generic Relations

We now present relationships between the parameters of point sets, which will be usefull in establishing our upper and lower bounds.

Lemma 2.6. *For every point set $\mathcal{P}$, source s, and $\ell \geqslant \ell^*$:*

$$0 < \ell^* \leqslant \rho^* \leqslant \xi_\ell \leqslant \xi_{\ell^*} \leqslant n \cdot \ell^* .$$

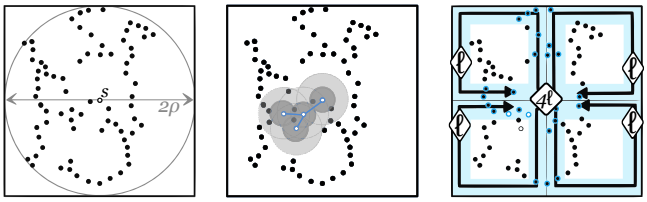
Lemma 2.7. *For any $r \geqslant \ell \geqslant 1$ we have:*

$$O(r + \ell^2(\log \min\{\ell, r/\ell\} + \log \frac{r}{\ell^{3/2}})) \subseteq O(r + \ell^2 \log(r/\ell)).$$

Lemma 2.8. *Let us consider a point set $(\mathcal{P}, s)$. For any $\ell \geqslant \ell^*$, we have $\xi_\ell \in [\rho^*, \frac{12\rho^{*2}}{\ell}]$, and for any position $p \in \mathcal{P}$, there exists a path from s to p in the ℓ-disk graph which is at most $1 + \frac{2\xi_\ell}{\ell}$-hops long.*

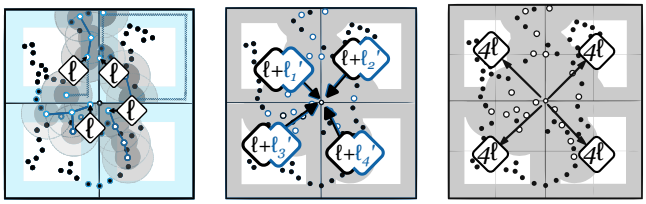
3 Without Energy Constraint

3.1 Algorithm $\mathcal{A}_{Separator}$



(a) Initialization. (b) DFSampling from s to get a sample of size 4ℓ. (c) Exploration of 4 separators by teams of ℓ robots.

Figure 1: First phases of $\mathcal{A}_{Separator}$; • sleeping robots; ∘ awake robots; ◉ sleeping seeds; ○ awake seeds.



(a) Recruitment: DFSampling starting from $\mathcal{X}_i \subset \text{sep}(\mathcal{S}_i)$. (b) $\mathcal{T}_i$ merge at the center of $\mathcal{S}$ and share their variables. (c) Next round: four team of 4ℓ move to a sub-square.

Figure 2: Recruitment and Reorganization phases of $\mathcal{A}_{Separator}$; ○ recruited robots; explored area is grayed.

Algorithm $\mathcal{A}_{Separator}$ is based on a divide-and-conquer strategy where robots are organized in teams. As shown by a matching lower bound, the algorithm has optimal makespan. We describe it with six phases: Initialization, Partition, Exploration, Recruitment, Reorganization and Termination.

Having first awaken 4ℓ robots in the square of center s and width 2ρ (Initialization Phase presented in Figure 1a and 1b). we divide the square into 4 sub-squares, and send a team of ℓ robots in every sub-square (Partition Phase). During the Exploration Phase, each team collectively explores the separator of their sub-square and stores seeds in $\mathcal{X}$, that are the initial positions of robots (sleeping or already awake) belonging to the separator (Figure 1c). During the Recruitment Phase, each team wakes up new robots in its sub-square, such that these new robots *plus robots currently exploring* whose initial position is in that sub-square, are 4ℓ (Figure 2a). These 4ℓ robots are *recruited* to define a new team used in the next round.

Finally, during phase Reorganization, all 4 teams meet in the center of the square so robots can team up with robots initially located to the same sub-square. Each team thus formed go into its corresponding sub-square, and repeat the process until all robots have been awakened (Figures 2b, 2c). Recruitment Phase relies on function DFSampling presented in Section 2.4. Lemma 5.1 guarantees that if DFSampling returns a set with less than 4ℓ positions, then every robot located in the sampled region has been discovered (but not necessarily awakened). This justifies the Termination phase where awaken robots wake up the remaining sleeping robots with a centralized algorithm. Figure 3 presents a more detailed description of $\mathcal{A}_{Separator}$.

$\mathcal{A}_{Separator}$

(1) **Round** 0: **Initialization** and **Recruitment**
$\mathcal{S} \leftarrow$ square of width $R = 2\rho$ and centered at the source robot s.
$\mathcal{T} \leftarrow \{s\}$; $\mathcal{X} \leftarrow \{p_s\}$.
$\mathcal{T}$ recruits up to $4\ell - 1$ new robots using DFSampling$(\mathcal{S}, \mathcal{X})$.
Move $\mathcal{T}$ to the center of $\mathcal{S}$.
(2) **Round** $k \geqslant 1$ for a team $\mathcal{T}$ in square $\mathcal{S}$:
(i) Termination
If $|\mathcal{T}| < 4\ell$: do a centralized awakening of sleeping robots in $\mathcal{S}$ and terminate.
(ii) Partition
Partition $\mathcal{S}$ (resp. $\mathcal{T}$) into 4 sub-squares $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ (resp. 4 teams $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ of ℓ robots each).
Each team $\mathcal{T}_i$ performs in parallel:
(iii) Exploration
Using 4 times Explore, collectively explore sep$(\mathcal{S}_i)$.
Save in $\mathcal{X}_i$ the set of positions of sep$(\mathcal{S}_i)$ where a robot was found asleep, and those where a robot has already been awakened.
(iv) Recruitment
$\mathcal{T}_i$ recruits ℓ_i' robots by using DFSampling$(\mathcal{S}_i, \mathcal{X}_i)$.
Move $\mathcal{T}_i$ to the center of $\mathcal{S}$.
(v) Reorganization
Wait until the four teams $\mathcal{T}_i$ can merge and share their variables.
Reorganize robots in teams $\mathcal{T}_i'$ of size ℓ_i' by square of origin $\mathcal{S}_i$.
For each team $\mathcal{T}_i'$, do in parallel:
- Move $\mathcal{T}_i'$ to the center of $\mathcal{S}_i$.
- Go to Round $k+1$ with team $\mathcal{T}_i'$ and square $\mathcal{S}_i$.

Figure 3: Description of $\mathcal{A}_{Separator}$.

Theorem 3.1. *$\mathcal{A}_{Separator}$ solves the dFTP, given every admissible tuple (ℓ, ρ, n), with a makespan $O(\rho + \ell^2 \log(\rho/\ell))$.*

3.2 Proof of Theorem 3.1

Let us first introduce some definitions. For $k \geqslant 1$, we denote by $\mathcal{T}^{(k)}$ and $\mathcal{S}^{(k)}$ a team executing DFSampling at Round k, and a region in which it is executed. Note that for a given $k > 1$, several squares are treated in parallel so neither $\mathcal{S}^{(k)}$ nor $\mathcal{T}^{(k)}$ are unique. We denote by $R^{(k)} = \frac{2\rho}{2^{k-1}}$ the width of $\mathcal{S}^{(k)}$. Round $k \geqslant 1$ is a *terminating* Round for team $\mathcal{T}^{(k)}$ in region $\mathcal{S}^{(k)}$ if $|\mathcal{T}^{(k)}| < 4\ell$, otherwise it is a *partitioning* Round.

Lemma 3.2 (Number of Rounds). *The number of rounds before $\mathcal{A}_{Separator}$ terminates is 1 if $\rho^* \leqslant \frac{\ell^{3/2}}{8}$, and $O(\log \frac{\rho}{\ell^{3/2}})$ otherwise.*

Proof. Set $R = 2\rho^*$. If $R \leqslant \frac{\ell^{3/2}}{4}$ then according to Lemma 2.5, any ℓ-sampling has size at most $\frac{16R^2}{4\pi\ell^2} \leqslant 4\ell/\pi < 4l$, and algorithm $\mathcal{A}_{Separator}$ stops in Round 1. Let us now prove the generic bound.

Let us express a condition on k such that all the executions of DFSampling in regions $\mathcal{S}^{(k)}$ of width $R^{(k)} = \frac{2\rho}{2^{k-1}}$ output an ℓ-sampling $\mathcal{T}'^{(k)}$ with size less than 4ℓ. This implies that $\mathcal{A}_{Separator}$ ends at the beginning of Round $k+1$, during the Termination phase.

By Lemma 2.5, $|\mathcal{T}'^{(k)}| \leqslant \frac{16(\frac{2\rho}{2^{k-1}})^2}{\pi\ell^2} = \frac{256\rho^2}{2^{2k}\pi\ell^2}$. It is therefore sufficient to have $\frac{256\rho^2}{2^{2k}\pi\ell^2} < 4\ell$, which is equivalent to $2^{2k} > \frac{64\rho^2}{\pi\ell^3}$, and finally to $k > \log(8\rho/\sqrt{\pi}\ell^{3/2}) \in \Theta(\log \rho/\ell^{3/2})$. □

Lemma 3.3 (Duration of Rounds). *For $k \geqslant 1$, the duration of Round k is in $O(R^{(k)} + \ell^2)$.*

Proof. We show that:

- a terminating Round has a duration $O(R^{(k)})$;
- a partitioning Round has a duration $O(R^{(k)} + \ell^2)$.

Let us first suppose that Round k is a terminating Round for $\mathcal{T}^{(k)}$ in $\mathcal{S}^{(k)}$, i.e., $|\mathcal{T}^{(k)}| < 4\ell$. Round k consists in the execution of a centralized algorithm to wake up sleeping robots in $\mathcal{S}^{(k)}$. The smallest disk containing $\mathcal{S}^{(k)}$ is of radius $(1/\sqrt{2})R^{(k)}$. By Lemma 2.2, one robot of $\mathcal{T}^{(k)}$ can wake up any set of known robots sleeping in this disk with a makespan in $O(R^{(k)})$.

We now suppose that Round k is a partitioning Round for $\mathcal{T}^{(k)}$ in $\mathcal{S}^{(k)}$, i.e., $|\mathcal{T}^{(k)}| \geqslant 4\ell$. It consists of having four teams of ℓ robots exploring and recruiting within sub-squares of $\mathcal{S}^{(k)}$ of width $R^{(k+1)}$. The separator of a sub-square can be decomposed into 4 rectangles of dimension $\ell \times R^{(k+1)}$. By Corollary 2.1 a team of ℓ robots can explore each rectangle in $O(\frac{\ell \times R^{(k+1)}}{\ell} + \ell + R^{(k+1)}) = O(R^{(k+1)} + \ell)$. The recruitment is done by DFSampling from a set of positions $\mathcal{X}$ in $\text{sep}(\mathcal{S}^{(k)})$. From Lemma 5.1 the ℓ-sampling of $\mathcal{S}^{(k+1)}$ with a team of ℓ robot is done in time $O(R^{(k+1)}) + \ell^2)$. Recall that $R^{(k+1)} = \frac{1}{2}R^{(k)}$, we get that the duration of a partitioning Round k is $O(R^{(k)} + \ell^2)$. □

Lemma 3.4 (Validity of $\mathcal{A}_{Separator}$). *For every admissible tuple (ℓ, ρ, n) and for any n-point set $\mathcal{P}$ and source s such that $\ell^* \leqslant \ell$ and $\rho^* \leqslant \rho$, the execution of $\mathcal{A}_{Separator}$ with source s and inputs (ℓ, ρ, n) on $\mathcal{P}$ eventually wakes up all robots.*

Proof. Consider an execution of $\mathcal{A}_{Separator}$ on $\mathcal{P}$. According to Lemma 3.2 an execution terminates after a finite number of Rounds. We show that when the execution terminates, any robot $r \in \mathcal{R}$ initially located at $p \in \mathcal{P}$ is awakened. Let us consider $r \in \mathcal{R}$ and the largest k such that at Round k there is a square $\mathcal{S}^{(k)}$ containing p. Let us prove that r is awake at the end of Round k.

Let us first prove by contradiction that Round k is a terminating Round for $\mathcal{T}^{(k)}$ on $\mathcal{S}^{(k)}$. If not, i.e., it is a partitioning Round, then for $i \in [1,4]$ there is an execution of Round $k+1$ by $\mathcal{T}_i'^{(k+1)} \neq \emptyset$ on sub-square $\mathcal{S}_i^{(k)}$. Since robots are assigned to teams $\mathcal{T}_i'$ based on their initial position, if $\mathcal{T}_i'$ is empty then $\mathcal{P} \cap \text{sep}(\mathcal{S}_i^{(k)}) = \emptyset$, by Lemma 2.4, $\mathcal{S}_i^{(k)}$ is empty and does not contain p. Hence if Round k is a partitioning Round then there is an execution of Round $k+1$ on a region $\mathcal{S}^{(k+1)}$ containing p, which breaks the assumption of k being maximal.

If Round k is terminating, by definition of $\mathcal{A}_{Separator}$, team $\mathcal{T}^{(k)}$ contains less than 4ℓ robots. It implies that during the recruitment phase of Round $k-1$, DFSampling in $\mathcal{S}^{(k)}$ has stopped before constituting a team of 4ℓ robots. Lemma 5.1 guarantees that $\mathcal{S}^{(k)}$ is covered by $\mathcal{T}^k$ i.e., each position of $\mathcal{P} \cap \mathcal{S}^{(k)}$ is known by $\mathcal{T}^{(k)}$ at the beginning of Round k. Therefore r must be awakened when $\mathcal{T}^{(k)}$ operates a centralized awakening of sleeping robots of $\mathcal{S}^{(k)}$. □

Lemma 3.5 (Makespan of $\mathcal{A}_{Separator}$). *For every admissible tuple (ℓ, ρ, n) and for any n-point set $\mathcal{P}$ and source s such that $\ell^* \leqslant \ell$ and $\rho^* \leqslant \rho$, the execution of $\mathcal{A}_{Separator}$ with source s and inputs (ℓ, ρ, n) on $\mathcal{P}$ terminates in time $O(\rho + \ell^2 \log(\rho/\ell))$*

Proof. Since $\mathcal{A}_{Separator}$ is decomposed into rounds, the proof is done by bounding the duration of a Round k, denoted t_k, and the number of rounds until $\mathcal{A}_{Separator}$ terminates.

The duration of Round 0 is the duration of recruiting a team $\mathcal{T}$ of size up to 4ℓ within a square of width $2\rho^*$. The recruitment is done by computing a ℓ-sampling of the square with DFSampling. According to Lemma 2.5 there is at most $32\rho^{*2}/(\pi\ell^2)$ points in a ℓ-sampling of a square of width $2\rho^*$, therefore $|\mathcal{T}| = \min(4\ell, \frac{32\rho^{*2}}{\pi\ell^2})$. Lemma 5.1 guarantees that the time of DFSampling depends of the size of the output team. Hence the duration of Round 0 is:

$$t_0 \in O(\ell^2 \log(\min\{\ell, \rho^*/\ell\})) \subseteq O(\ell^2 \log(\min\{\ell, \rho/\ell\})). \quad (1)$$

A terminating round only takes time $O(R^{(k)}) \subseteq O(\rho)$ to move toward the center of the sub-square and wake-up remaining sleeping robots discovered in the DFSampling execution. Let us focus on partitioning rounds. We showed in Lemma 3.3 that for $k \geqslant 1, t_k \in O(R^{(k)} + \ell^2)$.

Let k_{max} be the maximum number of Rounds. According to Lemma 3.2, $k_{max} \in O(\log \frac{\rho}{\ell^{3/2}})$. Let us now estimate the total duration of the rounds posterior to 0:

$$\sum_{k=1}^{k_{max}} t_k \in \sum_{k=1}^{k_{max}} O(R^{(k)} + \ell^2) \subseteq k_{max} O(\ell^2) + \sum_{k=1}^{k_{max}} O(\frac{2\rho}{2^{k-1}})$$
$$\in O(\rho) + O(\log \rho/\ell^{3/2}) O(\ell^2) \subseteq O(\rho + \ell^2 \log \rho/\ell^{3/2}).$$

Finally, using Lemma 2.7, we obtain:

$$T_{\mathcal{A}_{Separator}} = \sum_{k=0}^{k_{max}} t_k = t_0 + \sum_{k=1}^{k_{max}} t_k$$
$$\in O(\ell^2 \log(\min\{\ell, \rho/\ell\})) + O(\rho + \ell^2 \log \rho/\ell^{3/2})$$
$$\in O(\rho + \ell^2 \log(\rho/\ell)).$$

□

3.3 Lower Bound Without Energy Constraint

We now provide a lower bound on the makespan of any algorithm solving the dFTP.

Theorem 3.6 (Lower Bound without energy constraint). *For every admissible tuple (ℓ, ρ, n) and algorithm $\mathcal{A}$ solving the dFTP, there exists an n-point set $\mathcal{P}$ and a source s such that $\ell^* \leqslant \ell$ and $\rho^* \leqslant \rho$ such that the makespan of the execution of $\mathcal{A}$ with source s and inputs (ℓ, ρ, n) on $\mathcal{P}$ is $\Omega(\rho + \ell^2 \log(\rho/\ell))$.*

The rest of this section constitutes the proof of Theorem 3.6. Given $r \geqslant 0$ and $p \in \mathbb{R}^2$, we denote by $B_p(r)$ the disk of center p and radius r.

Let us consider an algorithm $\mathcal{A}$ solving the dFTP, and an admissible tuple (ℓ, ρ, n). To prove our lower bound, we first define some disjoint regions D_c of $B_{0,0}(\rho)$ as pictured in Figure 4a. Each region has area $\Theta(\ell^2)$, and according to Lemma 3.7, there are $\Theta(\rho^2/\ell^2)$ such regions. The general idea is to place one sleeping robot in each region, depending on the behavior of $\mathcal{A}$, in a way that guarantees that awake robots have to visit the entire region before they find the sleeping robot in it. We state in Lemma 3.8 that the set of points we propose is ℓ-connected, which makes the construction valid.

Centers and Connectivity. We define $C = \{(x, y) \in (\frac{\ell}{2}\mathbb{Z})^2 \mid \sqrt{x^2 + y^2} \leqslant \rho - \frac{\ell}{4}\}$ the vertices of the $\frac{\ell}{2}$-grid, restricted to the disk of center $(0, 0)$ and radius $\rho - \frac{\ell}{4}$. We say that two elements (x, y) and (x', y') in C are *adjacent* if $x' = x \wedge y' = y \pm \frac{\ell}{2}$ or $x' = x \pm \frac{\ell}{2} \wedge y' = y$. A subset $C \subseteq C$ is *connected* if for any $c, c' \in C$, there exists a path of adjacent elements of C with extremities c and c'. We also define $C^* = C \setminus \{(0, 0)\}$ and $m = \min(n, |C^*|)$. Note that, since (ℓ, ρ, n) is admissible, and according to Lemma 3.7, we have $m \geqslant \rho/\ell$. We are now going to prove a lower bound of $\Omega(\ell^2 \log m) \subseteq \Omega(\ell^2 \log \rho/\ell)$.

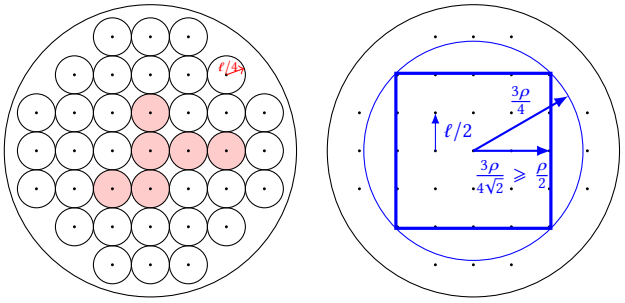
Lemma 3.7. $|C| \geqslant 1 + \rho^2/\ell^2$.

We denote by C_m some subset of C^* with m elements such that $C_m \cup \{(0, 0)\}$ is connected, and contains at least the elements $\{(0, \frac{\ell}{2}), (0, 2\frac{\ell}{2}), \ldots, (0, \lfloor \rho/\ell \rfloor \frac{\ell}{2})\}$. Note that this is always feasible because, on the one hand, $m \geqslant \lfloor \rho/\ell \rfloor$, and on the other hand, $\lfloor \rho/\ell \rfloor \frac{\ell}{2} \leqslant \rho/2 \leqslant \rho - \frac{\ell}{4}$, so all these points are actually in C^*.

Lemma 3.8. *For any adjacent points $c, c' \in C_m$, for any two points $(p, p') \in D_c \times D_{c'}$, we have $|pp'| \leqslant \ell$.*

Disks, ℓ-connectivity. For any $c = (x, y) \in C_m$, let us define the disk of center c by $D_c = B_c(\ell/4)$, of area $\frac{\pi\ell^2}{16}$. Let us also define $\mathcal{D}_m = \cup_{c \in C_m} D_c$ and $\mathcal{D}^* = \cup_{c \in C^*} D_c$. Note that different disks in $\mathcal{D}_m$ are pairwise disjoints (except one single point), that $\mathcal{D}_m \subset B_{(0,0)}(\rho)$, and that the area of $\mathcal{D}_m$ is $|C_m|\frac{\pi\ell^2}{16}$.

Construction of the Set $\mathcal{P}(\mathcal{A})$. Let us first suppose that $n \leqslant |C^*|$. The construction of the set of initial positions $\mathcal{P}(\mathcal{A})$ depends on the considered algorithm $\mathcal{A}$. The process consists in placing exactly one robot per disk $D_c \in \mathcal{D}_m$, at position p_c. This construction guarantees that the instance $(\mathcal{P}, s)$ is consistent with the tuple (ℓ, ρ, n). In particular, the set is actually ℓ-connected according to Lemma 3.8 and because C_m is connected. Given one disk $D_c \in \mathcal{D}_m$, the exact localisation p_c is defined as the last position of S_c to be explored by the previously awakened robots, under the execution of the algorithm. In other words, the algorithm must integrally explore S_c before it discovers the new robot in it. If $n > |C^*|$, then the construction is similar for the first $|C^*| - 1$ positions. The initial localization of the remaining robots can be in any arbitrary small disk of radius ε included in some area of $\mathcal{D}_m$ that has not been discovered yet.



(a) Scheme of the general construction of Theorem 3.6. $\mathcal{D}^*$ colored in red.

(b) Scheme of the proof of Lemma 3.7. S drawn in blue.

Figure 4: Proof of the Lower Bound.

Proof of the Lower Bound $\Omega(\ell^2 \log m)$. We denote by $\mathcal{R}_{\mathcal{A}}(t)$ the set of awake robots at time t, and given the set of initial positions $\mathcal{P} = \mathcal{P}(\mathcal{A})$, we denote by $\mathcal{D}_{\mathcal{P}}(t) \subseteq \mathcal{D}_m$ the set of points of $\mathcal{D}_m$ that have been discovered at time t or before. More formally, $\mathcal{D}_{\mathcal{P}}(t)$ is the set of points $p \in \mathcal{D}_m$ such that $\exists t' \leqslant t, \exists r \in \mathcal{R}_{\mathcal{A}}(t') : |p_r(t')p| \leqslant 1$.

We denote by $\|\mathcal{D}_{\mathcal{P}}(t)\|$ the area of $\mathcal{D}_{\mathcal{P}}(t)$. Finally, let us define, $\forall i \in [0, m], t_i = \inf\{t \geqslant 0 \mid \|\mathcal{D}_{\mathcal{P}}(t)\| \geqslant i\pi\ell^2/16\}$. By construction of $\mathcal{P}(\mathcal{A})$ we have $\forall i, \forall t < t_i, |\mathcal{R}_{\mathcal{A}}(t_i)| \leqslant i$. Since robots have a field of view of radius 1, they discover an area of amplitude 2 along an unitary move, which means that an awake robot exploring during t' units of time can discover an area of at most $2t'$. Therefore we have $\forall i, \forall t \geqslant t_i, \|\mathcal{D}_{\mathcal{P}}(t)\| - \|\mathcal{D}_{\mathcal{P}}(t_i)\| \leqslant 2(t - t_i)|\mathcal{R}_{\mathcal{A}}(t)|$. Furthermore, $\|\mathcal{D}_{\mathcal{P}}(t_{i+1})\| - \|\mathcal{D}_{\mathcal{P}}(t_i)\| = \frac{\pi\ell^2}{16}$. By having t tend to t_{i+1} by inferior values, we obtain: $\forall i \geqslant 0, \|\mathcal{D}_{\mathcal{P}}(t_{i+1})\| - \|\mathcal{D}_{\mathcal{P}}(t_i)\| \leqslant 2(t_{i+1} - t_i) \times (i + 1)$.

Therefore by adding this telescopic sum, we obtain:

$$t_m \geqslant \frac{\pi\ell^2}{32} \sum_{i=1}^{m} i \geqslant \frac{\pi\ell^2}{32} \ln(m + 1) \in \Omega(\ell^2 \log m)$$

Conclusion. Since $n \geqslant \frac{\rho}{\ell}$ and $|C^*| \geqslant \rho^2/\ell^2$, and $m = \min(n, |C^*|)$, we have obtained a lower bound of $\Omega(\ell^2 \log \frac{\rho}{\ell})$. Furthermore, by construction of C^m, and by definition of $\mathcal{P}$, robots have explore all points of $D_{(0, \lfloor \rho/\ell \rfloor \frac{\ell}{2})}$, which means that there exists a path from $(0, 0)$ to a point at distance 1 of $(0, \lfloor \rho/\ell \rfloor \frac{\ell}{2})$. Such a path has a length at least $\lfloor \rho/\ell \rfloor \frac{\ell}{2} - 1 \geqslant \frac{\rho}{2\ell}\frac{\ell}{2} - 1 \geqslant \rho/4 - 1 \in \Omega(\rho)$. We therefore have shown that the makespan of $\mathcal{A}$ on that set of point is $\Omega(\rho + \ell^2 \log \frac{\rho}{\ell})$.

4 With Energy Constraint

The construction used in the proof of Theorem 3.6 can be adapted to obtain Theorem 4.1.

Theorem 4.1 (Lower Bound on the energy budget). *For every admissible tuple (ℓ, ρ, n) and algorithm $\mathcal{A}$ with energy budget $B < \pi(\ell^2 - 1)/2$, there exists an n-point set $\mathcal{P}$ and a source s such that*

$\ell^* \leqslant \ell$ and $\rho^* \leqslant \rho$ and the execution of $\mathcal{A}$ on $\mathcal{P}$ does not wake-up any robot.

4.1 Energy-Optimal Algorithm

We use a Breadth First Search based strategy to define $\mathcal{A}_{Grid}$, an algorithm that takes as input the parameter $\ell \geqslant \ell^*$. Basically, $\mathcal{A}_{Grid}$ works as follows: the plane is partitioned into squares of width 2ℓ centered at positions $\{(2k\ell, 2k'\ell) \mid (k,k') \in \mathbb{Z}^2\}$. We wake-up the square $\mathcal{S}$ containing s in time $t(\mathcal{S}) = R^2 + (10+\sqrt{2})R$ (Corollary 2.3). Then every square containing a new awake robot tries to wake up the 8 adjacent squares of the square ordered in a counter-clockwise order. Figure 5 presents a detailed presentation of $\mathcal{A}_{Grid}$.

Let $t(\ell)$ be an upper bound on the time required for the exploration and centralized awakening of a square region of width 2ℓ by one robot. By Corollary 2.3, $t(\ell) \in O(\ell^2)$.

$\mathcal{A}_{Grid}$

- *Round* 0 - *Initialization*:
  $\mathcal{S} \leftarrow$ square of width $R = 2\ell$, centered at the source s
  Explore and wake-up $\mathcal{S}$
- *Round* $k \geqslant 1$ *starting at time* $t_k = t(\ell) + 8(k-1)(t(\ell) + \sqrt{2}R)$:
  For every robot r awakened in Round $k-1$ do in parallel:
  $\mathcal{S} \leftarrow$ the square containing r
  For $i \in [1..8]$ do
  (1) move to the lower-left corner of the i-th adjacent squares $\mathcal{S}_i$ of $\mathcal{S}$
  (2) Wait until time $t_k + (t(\ell) + \sqrt{2}R)i$
  (3) Explore and Wake-up $\mathcal{S}_i$

Figure 5: Description of $\mathcal{A}_{Grid}$.

Theorem 4.2. *$\mathcal{A}_{Grid}$ solves the dFTP, given every admissible tuple (ℓ, ρ, n), with an energy budget of $O(\ell^2)$ and with a makespan of $O(\ell \cdot \xi_\ell)$.*

Proof. The duration of Round 0 is $t(\ell)$. Then, in Round $k \geqslant 1$, we have to wake-up at most 8 squares in time $8t(\ell)$. Since every square are adjacent, it takes at most $\sqrt{2}R$ to go the next adjacent square to wake-up. In total, every round takes $O(\ell^2)$. Every robot awakened in Round k participates to Round $k+1$ but stops do at the end of Round $k+1$. Thus it only has to use $O(\ell^2)$ energy.

Let us now upper bound the number of rounds to wake up some robot of $\mathcal{P}$ (which proves that $\mathcal{A}_{Grid}$ solves the dFTP). Given any robot r located at position p, let us consider a shortest path from s to r in the ℓ-disk graph of $(\mathcal{P}, s)$ minimizing the number of hops and the sequence of squares $S = s_1, \ldots, s_k$ crossed by this path. Since the path can cross several times a same square and every robot of a given square are awakened in the same round, the number of rounds required to wake up the sequence of squares is smaller or equal to the number of hops of the shortest path. From Lemma 2.8, we know that there exists a path of at most $2\xi_\ell/\ell$ hops implying the same upper bound for the number of rounds. Since every round takes a time $O(\ell^2)$, the makespan is $O(\ell \cdot \xi_\ell)$. Furthermore since every awakened robot in Round k only moves in Round k and $k+1$ and stops, robots only need $O(\ell^2)$ energy. □

4.2 Makespan-Optimal Algorithm

The next algorithm, $\mathcal{A}_{Wave}$, is an adaptation of $\mathcal{A}_{Grid}$. Roughly speaking, there are two changes: (1) the squares are now of width $8\ell^2 \log_2 \ell$; and (2) instead of using a simple process to explore and wake up a square, we use $\mathcal{A}_{Separator}$ starting from a team of 4ℓ robots for Rounds $k \geqslant 1$ and from the source s for Round 0. Algorithm $\mathcal{A}_{Wave}$ is detailed in Figure 6.

Let $t(R)$ be an upper bound on the time required for $\mathcal{A}_{Separator}$ to wake up all robots of a square of width R starting from a team of size 4ℓ. By Theorem 3.1, $t(R) \in \Theta(R + \ell^2 \log \ell)$.

$\mathcal{A}_{Wave}$

- *Round* 0 - *Initialization*:
  $\ell \leftarrow \max\{\ell, 4\}$
  $\mathcal{S} \leftarrow$ square of width $R = 8\ell^2 \log_2 \ell$, centered at the source s
  Wake-up $\mathcal{S}$ using $\mathcal{A}_{Separator}$
  If there is no robot in $\mathrm{sep}(\mathcal{S})$, $\mathcal{A}_{Wave}$ stops
- *Round* $k \geqslant 1$ *starting at time* $t_k = t(R) + 8(k-1)(t(R) + \sqrt{2}R)$, for every r awakened in Round $k-1$:
  $\mathcal{S}(r) \leftarrow$ square containing r
  Move r to the lower-left corner of $\mathcal{S}(r)$ to build team $\mathcal{T}_r$
  For every team $\mathcal{T}_r$ such that $|\mathcal{T}_r| \geqslant 4\ell$ do in parallel:
  For $i \in [1..8]$ do
  (1) move to the lower-left corner of the i-th adjacent squares $\mathcal{S}_i$ of $\mathcal{S}(r)$
  (2) Wait until time $t_k + (t(R) + \sqrt{2}R)i$
  (3) Wake-up $\mathcal{S}_i$ using $\mathcal{A}_{Separator}$ within $\mathcal{S}_i$ with $\mathcal{T}_r$ in Round k

Figure 6: Description of $\mathcal{A}_{Wave}$.

Theorem 4.3. *$\mathcal{A}_{Wave}$ solves the dFTP, given every admissible tuple (ℓ, ρ, n), with an energy budget of $O(\ell^2 \log \ell)$ and a makespan of $O(\xi_\ell + \ell^2 \log \xi_\ell/\ell)$.*

Proof. We first prove that $\mathcal{A}_{Separator}$ solves the dFTP with energy budget $O(\ell^2 \log \ell)$ and a makespan of $O(\xi_\ell + \ell^2 \log \ell)$, and explain in a second time how we obtain the announced makespan. Firstly, note that $t(R) \in \Theta(\ell^2 \log \ell)$.

To begin, every round takes a time at most $O(R) = O(\ell^2 \log \ell)$. The only robots that participates in Round k has been awakened only in Round $k+1$, it means that the energy budget required per robot is $O(\ell^2 \log \ell)$.

At Round 0, $\mathcal{A}_{Separator}$ wakes up every robot of the initial square $\mathcal{S}$. If there is a robot within $\mathrm{sep}(\mathcal{S})$, it means that $\mathcal{S}$ contains at least $\left\lfloor \frac{R/2-\ell}{\ell} \right\rfloor \geqslant \frac{R}{2\ell} - 2 = 4\ell \log_2 \ell - 2 \geqslant 4\ell$ robots since $\ell \geqslant 4$. Thus, $\mathcal{S}$ has enough robots to apply $\mathcal{A}_{Separator}$ in every adjacent squares at Round 1, and every robot within the 8 adjacent squares are awakened during Round 1.

Let us now prove that all robots are awakened by the algorithm by providing an upper bound on the number of rounds to wake up a robot r located outside the 9 central squares. Let G be the ℓ-disk graph of $\mathcal{P}$ and let $P = s, r_1, r_2 \ldots, r$ be a shortest path in G from s to r, and let $s, r_1, r_2, \ldots, r_{j'}$ be the maximal subpath of P of awake robots at the end of Round k. We now consider a robot r_j with $j < j'$ such that $R - 3\ell < d_\ell(r_j, r_{j'}) \leqslant R - 2\ell$. Such a robot exists since by definition of s, $d_\ell(s, r) > R$.

First, let us show that the subpath $P' = r_j, r_{j+1}, \ldots, r_{j'}$ is contained within $\mathcal{S}(r_{j'})$ and at most 2 squares simultaneously adjacent to $\mathcal{S}(r_{j'})$ and $\mathcal{S}(r_{j'+1})$. Let $\mathcal{S}$ be a square, adjacent to $\mathcal{S}(r_{j'})$ but not adjacent to $\mathcal{S}(r_{j'+1})$. The Euclidean distance between $r_{j'}$ and $\mathcal{S}$ is greater than $R - \ell$ since $r_{j'}$ is at distance at most ℓ from $\mathcal{S}(r_{j'+1})$. Thus P' cannot cross $\mathcal{S}$. Moreover, it is impossible that P' crosses two adjacent squares of $\mathcal{S}(r_{j'})$, $\mathcal{S}$ and $\mathcal{S}'$, such that $\mathcal{S}$ and $\mathcal{S}'$ are not adjacent because P' has a length strictly smaller then R. This guarantees the announced property on P'.

Secondly, let us show that $\mathcal{S}(r_{j'+1})$ is awakened at Round $k+1$. To have this guarantee, we need to have at least 4ℓ awakened at Round k in an adjacent square. Since P' is contained within at most 3 adjacent squares of $\mathcal{S}(r_{j'+1})$, the most populated adjacent square contains at least $\left\lfloor \frac{R-2\ell}{\ell} \right\rfloor /3 = \left\lfloor \frac{R}{\ell} \right\rfloor /3 - 2/3 \geqslant (R/\ell - 1)/3 - 2/3 = R/(3\ell) - 1$ awake robots. If $R = 8\ell^2 \log \ell$ and $\ell \geqslant 4$, then $R/(3\ell) - 1 \geqslant 16\ell/3 - 1 \geqslant 4\ell$.

Now, take the maximal subpath P'' of P of length smaller than $R - 2\ell$ but starting from $r_{j'+1}$. We can show similarly as before that this path is either within 3 adjacent squares awakened in the worst case in Rounds $k+1$, $k+2$ and $k+3$ or that this path ends in already awakened square. In any case, within 3 rounds, the length of the maximal subpath of P of awake robots at the end of Round $k+3$ has increased of at least $R - 3\ell > 5\ell^2 \log \ell$ units. Thus the number of rounds to wake up the robots of highest eccentricity takes $O(\xi_\ell/\ell^2 \log \ell)$ rounds. The makespan is then bounded by $O(\xi_\ell)$.

Finally, let us be more precise on the bound on the makespan. We first consider the case where $\xi_\ell \leqslant \ell^{*3/2}/16 \leqslant \ell^{3/2}/16$. We immediately have $\xi_\ell < R/2$ and therefore $\mathcal{A}_{Wave}$ terminates at Round 0. But since $\xi_\ell \leqslant \ell^{3/2}/16$, we have by Lemma 2.6, $\rho^* \leqslant \ell^{3/2}/16$ and so, by Lemma 3.2, $\mathcal{A}_{Separator}$ terminates at Round 0. In that case, as stated by Equation 1 in the proof of Lemma 3.5, the makespan of $\mathcal{A}_{Wave}$ is $t_0 \in O(\ell^2 \log(\min\{\ell, \rho^*/\ell\}))$. Since by Lemma 2.6, $\rho^*/\ell \leqslant \xi_\ell/\ell$ and by Lemma 2.8, $\xi_\ell/\ell \leqslant 12\rho^{*2}/\ell^2$, we obtain an overall complexity of $\mathcal{A}_{Wave}$ is in $O(\ell^2 \log(\min\{\ell, \xi_\ell/\ell\}))$.

Otherwise we have $\xi_\ell \geqslant \ell^{3/2}/16$ and equivalently, the relations $\xi_\ell/\ell \geqslant \sqrt{\ell}/16$ and $\min\{\sqrt{\ell}/16, \xi_\ell/\ell\} = \sqrt{\ell}/16$.
Hence $O(\ell^2 \log \ell) = O(\ell^2 \log(\min\{\ell, \xi_\ell/\ell\}))$. This guarantees an overall complexity in $O(\xi_\ell + \ell^2 \log \min\{\ell, \xi_\ell/\ell\})$ for $\mathcal{A}_{Wave}$.

To summarize, we have obtained that $\mathcal{A}_{Wave}$ has an overall complexity in $O(\xi_\ell + \ell^2 \log(\min\{\ell, \xi_\ell/\ell\}))$. By Lemma 2.7, we conclude that $\mathcal{A}_{Wave}$ has a makespan in $O(\xi_\ell + \ell^2 \log \xi_\ell/\ell)$. □

4.3 Lower Bound with Energy Constraint

We present a lower bound for a given of range of values for ξ.

Theorem 4.4 (Lower Bound for energy constrained). *For every admissible tuple (ℓ, ρ, n), for every $B > \ell$, and for every $\xi \in [\rho, \min\{n\ell - \rho/3, \lfloor \rho^2/(2(B+1)) + 1 \rfloor\}]$, there exists an n-points set $\mathcal{P}$ and a source s of connectivity threshold ℓ, radius ρ, and ℓ-eccentricity $\xi_\ell = \xi$ such that the makespan of any algorithm $\mathcal{A}$ solving the dFTP for $(\mathcal{P}, s)$ given (ℓ, ρ, n) and energy budget B, is $\Omega(\xi + \ell^2 \log(\xi/\ell))$.*

5 Details on the Distributed ℓ-Sampling

We describe more formally the algorithm DFSampling. In a region $\mathcal{S}$ a team $\mathcal{T}$ computes an ℓ-sampling of $\mathcal{S}$, starting from a given set of seeds $\mathcal{X}$. If robots have been awakened previously in $\mathcal{S}$, we denote by A the set of their initial positions. A is assumed to be known by $\mathcal{T}$. The output is a set of points $\mathcal{P}'$ that is a ℓ-sampling of $\mathcal{S}$. In practice, we use DFSampling to recruit a team $\mathcal{T}'$ of at most 4ℓ robots identified by their positions $\mathcal{P}'$.

Let us start by defining a function Sort($\mathcal{X}$) to order positions of seeds $\mathcal{X} = \{X_i\}$ whenever $\mathcal{X} > 1$. Seeds are ordered with respect to a projection on the borders of $\mathcal{S}$ in the clockwise order around the center of square $\mathcal{S}$. More precisely, a seed X_i is projected to the closest point of the border of $\mathcal{S}$ breaking tie by choosing the first projected point in the clockwise order.

DFSampling

(1) $\mathcal{P}' = \emptyset$. Positions $\mathcal{X} = [X_1, X_2, \ldots, X_j]$ are ordered with Sort($\mathcal{X}$). Set $i = 1$.
(2) While $|\mathcal{P}'| < 4\ell$ and $i <= j$, adds X_i to $\mathcal{P}'$. $\mathcal{T}$ performs a DFS traversal of the 2ℓ-disk graph induced by $\mathcal{P} \cap \mathcal{S}$, starting by X_i:
  (a) When a position p is added to $\mathcal{P}'$, $\mathcal{T}$ moves to p and does Explore($B_{2\ell}(p), p$). If a sleeping robot was found at p, it is awakened and added to $\mathcal{T}$ beforehand.
  (b) Neighbors of p are known either from A or from exploration of $B_{2\ell}(p)$. A position $p' \in B_{2\ell}(p)$ is added to $\mathcal{P}'$ *if its distance to any point of $\mathcal{P}'$ is greater than ℓ.*
  (c) A stack keeps track of the neighbors of positions in $\mathcal{P}'$. $\mathcal{T}$ explores as far as possible along each branch and backtracks when no such p' was found around p.
  (d) $i \leftarrow$ the index of the next seed of $X_{i'}$ such that $B_{X_{i'}}(\ell)$ is not covered. Go to Step 2;
(3) Return $\mathcal{P}'$

Figure 7: Description of DFSampling.

We are now ready to prove the corresponding lemma:

Lemma 5.1 (DFSampling). *In a region $\mathcal{S}$ of width R containing a set of seeds $\mathcal{X} \subseteq \mathcal{P}$, a team $\mathcal{T}$ of robots knowing the awake robots of $\mathcal{S}$ can compute a set of positions $\mathcal{P}'$ being an ℓ-sampling of $\mathcal{P}$. The computation is done using DFSampling in time:*

$$O(\ell^2 \log(|\mathcal{P}'|)) \text{ if } |\mathcal{T}| = 1 \text{ and } \mathcal{X} = \{p_s\} \text{ and } R \geqslant 2\rho^*$$
$$O(R + \ell|\mathcal{P}'|) \text{ if } |\mathcal{T}| = \ell \text{ and } \mathcal{X} = \mathcal{P} \cap \text{sep}(\mathcal{S})$$

In both cases, either (1) $|\mathcal{P}'| = 4\ell$; or (2) $|\mathcal{P}'| < 4\ell$ and $\mathcal{S}$ is covered *by $\mathcal{P}'$.*

Proof. By construction, any p' added to $\mathcal{P}'$ is at distance greater than ℓ from any other point of P'. Inductively, any pair of positions

in $\mathcal{P}'$ has a pairwise distance greater than ℓ. Thus the output of DFSampling is an ℓ-sampling of $\mathcal{S}$. Let us now analyze the time required to obtain that ℓ-sampling.

The Depth First Search traversal uses three operations: find neighbors, move to a neighbor and backtrack if no new neighbor is added.

In the following $\mathcal{T}$ stands for the current team of robots. As soon as the robot team wakes up a sleeping robot, it is added to $\mathcal{T}$ so the team size can increase. Let us start with seed X_1. The time to find the list of potential neighbors in the 2ℓ-disk graph is the time to explore $B_p(2\ell)$. From Lemma 2.1, it takes a time in $O(\ell^2/|\mathcal{T}|)$. Thus exploring and moving to a neighbor takes $O(\ell^2/|\mathcal{T}| + \ell)$.

Let us first consider the case where the initial team $|\mathcal{T}| = 1$ and $\mathcal{X} = A = p_s$. Assume that at some point, k robots were awakened and added to $\mathcal{T}$. This happens as soon as there is a sleeping robot on a position p' that is added to $\mathcal{P}'$. The time to find and wake up these k robots is upper bounded by $\sum_{i=1}^{k} O(\ell^2/i) \in O(\ell^2 \log k)$.

Let us focus on the time dedicated to moves. For recruiting $k \geqslant 2$ robots, $\mathcal{T}$ either have to move forward to a neighbor at a distance at most 2ℓ, or backtrack to branch from a neighbor of a previously added robot. The amount of moves required to backtrack is at most $2k\ell$ units since the tree corresponding to the Depth First Search traversal has k edges of length at most 2ℓ. In total, the total time of backtracking is less than $2k\ell \leqslant 8\ell^2 \in O(\ell^2)$ since $k \leqslant 4\ell$. In total, it takes $O(\ell^2 \log k)$ for this first case.

The second case is whenever we start with a team of size ℓ and a set of seeds $\mathcal{X} = \mathcal{P} \cap \text{sep}(\mathcal{S})$. The exploration is now done collaboratively with ℓ robots. By Lemma 2.1, for any position $p' \in \mathcal{P}'$ the exploration phase takes $O(\ell)$. As in the first case, the different moves take $O(k\ell)$. Thus the total time of exploration for the recruitment of k robots takes $O(k\ell)$.

The second change is that the Depth First Search traversal starting from seed X_1 can stop before reaching 4ℓ robots. In this case the search goes on starting from the next seed. We have to add the time corresponding to the moves from one seed to another. Note that $\mathcal{T}$ starts a branch from at most 4ℓ positions $X_{i_1}, \ldots, X_{i_{4\ell}}$ of $\mathcal{X}$. Assume that $\mathcal{T}$ recruit k seeds among these 4ℓ positions. We rely on the fact that $\mathcal{X} \subseteq \text{sep}(\mathcal{S})$ and the ordering computed by $\textsc{Sort}(\mathcal{X})$ to bound the total duration of theses intermediates moves.

Let Y_i be the projected point of X_i on the border of $\mathcal{S}$. By definition of the ordering $\textsc{Sort}(\mathcal{X})$, the distance between two consecutive seeds X_{i_j} and $X_{i_{j+1}}$ is at most $|X_{i_j}Y_{i_j}|+|Y_{i_j}, Y_{i_{j+1}}|+|X_{i_{j+1}}Y_{i_{j+1}}|$ which is at most $2\ell + d_1(Y_{i_j}, Y_{i_{j+1}})$ where $d_1(p, p')$ is the distance in the L_1-norm for $p, p' \in \mathbb{R}^2$. Thus the distance travelled in the separator, $\sum_{j=1}^{k-1} |X_{i_j}X_{i_{j+1}}|$, is at most $\sum_{j=1}^{k-1} 2\ell + d_1(Y_{i_j}, Y_{i_{j+1}}) \leqslant 4R + 2k\ell$ since $\sum_{j=1}^{k-1} d_1(Y_{i_j}, Y_{i_{j+1}})$ is at most the perimeter of the square. To conclude, it takes $O(R + k\ell)$.

We now focus on properties (1) and (2). Assume that algorithm DFSampling ends. We show by contradiction that if $|\mathcal{P}'| < 4\ell$ then $\mathcal{S}$ is covered by $\mathcal{P}'$: $\forall p \in \mathcal{P} \cap \mathcal{S}, \exists p' \in \mathcal{P}'$ such that $p \in B_{p'}(2\ell)$.

Assume that there exists a point $p_0 \in \mathcal{P} \cap \mathcal{S}$ that is not covered by $\mathcal{P}'$. Since the connectivity threshold is less than ℓ, the ℓ-disk graph G of $\mathcal{P}$ is connected and any pair of points in $\mathcal{P}$ are linked by a path in G. In particular, there must exist a seed $p_x \in \mathcal{X}$ such that there is a path $b = (p_0, p_1, ..., p_x)$ in G entirely contains in $\mathcal{S}$: $\bigcup_{i=1}^{x} p_i \subset \mathcal{S}$. It is trivial if $\mathcal{P}$ is fully contained in $\mathcal{S}$ i.e., when $R \geqslant 2\rho^*$.

Otherwise, if there is in $\mathcal{P}$ at least one point q outside of $\mathcal{S}$ and $\mathcal{X} = \mathcal{P} \cap \text{sep}(\mathcal{S})$. From Lemma 2.4 any path from $p_0 \in \mathcal{S}^{in}$ to $q \in \mathcal{S}^{out}$ contains at least one position $p_x \in \text{sep}(\mathcal{S}) \cap \mathcal{X}$. Consider a path b from p_0 to p_x. By assumption, p_0 is not covered by $\mathcal{P}'$. Let p_i be the first position on the path such that p_i is not covered by $\mathcal{P}'$ and p_{i+1} is covered. By construction of $\mathcal{P}'$ if p_{i+1} is covered there is a position $p'_{i+1} \in \mathcal{P}'$ contains the close neighborhood of p_{i+1} in G i.e., $|p_{i+1}p'_{i+1}| \leqslant \ell$ or $p_{i+1} = p'_{i+1}$. This implies that $B_{p'_{i+1}}(2\ell)$ covered by P'. Since $|p_ip_{i+1}| \leqslant \ell$ and $|p_{i+1}p'_{i+1}| \leqslant \ell$ we get $|p_ip'_{i+1}| \leqslant 2\ell$ which conflicts with the assumption that p_i is not covered. □

6 Discussion

Although the input of our algorithms is (ℓ, ρ, n), it turns out that only the knowledge of ℓ as an upper bound of ℓ^* is required. The number of robots to awake n is never used.

Only $\mathcal{A}_{Separator}$ requires an upper bound ρ on ρ^*. We can easily get a constant approximation of ρ^* as follows: (1) we build a team of 4ℓ robots using DFSampling in time $O(\ell^2 \log \ell)$ and (2) we run explorations of the ℓ-separators squares of increasing width $\ell \cdot 2^i$ for $i = 1, 2, \ldots, k$ until we have an empty separator. Then we take $\rho = \ell \cdot 2^k$ which is a 2-approximation of ρ^*. The overcost is $O(\ell^2 \log \ell + \sum_i^k \ell 2^i) = O(\ell^2 \log \ell + \ell 2^{k+1}) = O(\ell^2 \log \ell + \rho)$. Either step (1) does not provide 4ℓ robots meaning that $\mathcal{P}$ is covered and that ρ^* is deduced from the discovered robots or (2) we have $\log \ell = O(\log(\rho^*/\ell))$. To conclude, the total overcost of computing a constant approximation of ρ^* from ℓ is of same order than $\mathcal{A}_{Separator}$.

We presented two algorithms that guarantee an optimal makespan under the hypothesis of knowing an upper bound on ℓ^*. $\mathcal{A}_{Separator}$ provides an optimal makespan in $\Theta(\rho^* + \ell^2 \log(\rho^*/\ell))$ without energy constraints, and $\mathcal{A}_{Wave}$ provides an optimal makespan of $\Theta(\xi_\ell + \ell^2 \log \xi_\ell/\ell)$ with an energy budget in $\Theta(\ell^2 \log \ell)$. Finally, $\mathcal{A}_{Grid}$ provides a less efficient makespan (apart under some specific relations between ℓ, ρ and ξ_ℓ), but requires an optimal energy budget of $\Theta(\ell^2)$.

Some questions remain open: can we get an optimal algorithm with only $\Theta(\ell^2)$ energy?

Moreover, our lower bound with energy constraints holds for a specific range of ξ, informally between ρ and $c \cdot (\rho/\ell)^2$ for some constant c, whereas we know that ξ can approach $c \cdot \rho^2/\ell$: can we obtain a lower bound for a broader range of value of ξ?

In our algorithms, robots communicate using rendez-vous. Is it possible to achieve efficient solution without communication?

Acknowledgments

Research supported in part by the French ANR projects ENEDISC (ANR-24-CE48-7768) and TEMPOGRAL (ANR-22-CE48-0001).

References

[1] Zachary Abel, Hugo A. Akitaya, and Yu Jingjin. 2017. Freeze Tag Awakening in 2D is NP-hard. In *27th Annual Fall Workshop on Computational Geometry (FWCG)* (Stony Brook University, NY, USA). https://www.ams.stonybrook.edu/~jsbm/fwcg17/proceedings.html

[2] Sharareh Alipour, Kajal Baghestani, Mahdis Mirzaei, and Soroush Sahraei. 2025. *Geometric Freeze-Tag Problem.* Technical Report 2412.19706 [cs.DC]. arXiv. https://doi.org/10.48550/arXiv.2412.19706 To appear in AAMAS'25.

[3] Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, Joseph S.B. Mitchell, and Martin Skutella. 2006. The Freeze-Tag Problem: How to Wake Up a Swarm of Robots. *Algorithmica* 46 (2006), 193–221. https://doi.org/10.1007/s00453-006-1206-1

[4] Esther M. Arkin, Michael A. Bender, Dongdong Ge, Simai He, and Joseph S.B. Mitchell. 2003. Improved Approximation Algorithms for the Freeze-Tag Problem. In *15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)* (San Diego, California, USA). ACM Press, 295–303. https://doi.org/10.1145/777412.777465

[5] Nicolas Bonichon, Arnaud Casteigts, Cyril Gavoille, and Nicolas Hanusse. 2024. Freeze-Tag in L_1 has Wake-up Time Five with Linear Complexity. In *38th International Symposium on Distributed Computing (DISC)* (Madrid, Spain) *(LIPIcs, Vol. 319)*, Dan Alistarh (Ed.). 9:1–9:16. https://doi.org/10.4230/LIPIcs.DISC.2024.9

[6] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Saeed Odak. 2024. Euclidean Freeze-Tag Problem on Plane. In *36th Canadian Conference on Computational Geometry (CCCG)* (St. Catharines, Ontario, Canada). 199–205. http://www.cccg.ca/

[7] Sébastien Bouchard, Yoann Dieudonné, Andrzej Pelc, and Franck Petit. 2020. Deterministic Treasure Hunt in the Plane with Angular Hints. *Algorithmica* 82, 11 (2020), 3250–3281. https://doi.org/10.1007/S00453-020-00724-4

[8] Josh Brunner and Julian Wellman. 2020. An Optimal Algorithm for Online Freeze-tag. In *10th International Conference Fun with Algorithms (FUN)* (La Maddalena, Italy) *(LIPIcs, Vol. 157)*. 8:1–11. https://doi.org/10.4230/LIPIcs.FUN.2021.8

[9] Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Barbara Geissmann, Daniel Graf, Arnaud Labourel, and Matúš Mihalák. 2020. Collaborative delivery with energy-constrained mobile robots. *Theoretical Computer Science* 810 (2020), 2–14. https://doi.org/10.1016/j.tcs.2017.04.018 Special issue on Structural Information and Communication Complexity.

[10] Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan. 2020. Sleeping is Efficient: MIS in O(1)-rounds Node-averaged Awake Complexity. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/3382734.3405718

[11] Shantanu Das. 2019. Graph Explorations with Mobile Agents. In *Distributed Computing by Mobile Entities*, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro (Eds.). Lecture Notes in Computer Science, Vol. 11340. Springer, Cham, Chapter 16, 403–422. https://doi.org/10.1007/978-3-030-11072-7_16

[12] Shantanu Das, Dariusz Dereniowski, and Przemyslaw Uznanski. 2024. Energy Constrained Depth First Search. *Algorithmica* 86, 12 (2024), 3759–3782. https://doi.org/10.1007/S00453-024-01275-8

[13] Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. 2006. Optimal constrained graph exploration. *ACM Trans. Algorithms* 2, 3 (July 2006), 380–402. https://doi.org/10.1145/1159892.1159897

[14] Miroslaw Dynia, Miroslaw Korzeniowski, and Christian Schindelhauer. 2006. Power-Aware Collective Tree Exploration. In *Architecture of Computing Systems - ARCS 2006, 19th International Conference, Frankfurt/Main, Germany, March 13-16, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3894)*, Werner Grass, Bernhard Sick, and Klaus Waldschmidt (Eds.). Springer, 341–351. https://doi.org/10.1007/11682127_24

[15] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sébastien Sereni. 2012. Collaborative search on the plane without communication. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, Darek Kowalski and Alessandro Panconesi (Eds.). ACM, 77–86. https://doi.org/10.1145/2332432.2332444

[16] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro (Eds.). 2019. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing.* Lecture Notes in Computer Science, Vol. 11340. Springer. https://doi.org/10.1007/978-3-030-11072-7

[17] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. 2006. Collective tree exploration. *Networks* 48, 3 (2006), 166–177. https://doi.org/10.1002/NET.20127

[18] G. Matthew Fricke, Joshua P. Hecker, Antonio D. Griego, Linh T. Tran, and Melanie E. Moses. 2016. A distributed deterministic spiral search algorithm for swarms. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016.* IEEE, 4430–4436. https://doi.org/10.1109/IROS.2016.7759652

[19] Cyril Gavoille, Nicolas Hanusse, Gabriel Le Bouder, and Taïssir Marcé. 2025. *Distributed Freeze Tag: a Sustainable Solution to Discover and Wake-up a Robot Swarm.* Technical Report 2503.22521 [cs.DS]. arXiv. https://doi.org/10.48550/arXiv.2503.22521 Full version of this paper.

[20] Mikael Hammar, Bengt J. Nilsson, and Mia Persson. 2006. The online freeze-tag problem. In *7th Latin American Symposium on Theoretical Informatics (LATIN)* (Valdivia, Chile) *(Lecture Notes in Computer Science, Vol. 3887)*. Springer, 569–579. https://doi.org/10.1007/11682462_53

[21] Christian Ortolf and Christian Schindelhauer. 2012. Online multi-robot exploration of grid graphs with rectangular obstacles. In *ACM Symposium on Parallelism in Algorithms and Architectures.* https://api.semanticscholar.org/CorpusID:14667301

[22] Ehsan Najafi Yazdia, Alireza Bagheri, Zahra Moezkarimia, and Hamidreza Keshavarz. 2015. An $O(1)$-approximation algorithm for the 2-dimensional geometric freeze-tag problem. *Inform. Process. Lett.* 115, 6-8 (June 2015), 618–622. https://doi.org/10.1016/j.ipl.2015.02.011