Collecting Information by Power-Aware Mobile Agents

Julian Anaya, Jérémie Chalopin, Jurek Czyzowicz, Arnaud Labourel, Andrzej Pelc, Yann Vaxès

Collecting information by power-aware mobile agents

Julian Anaya[1], Jérémie Chalopin[2], Jurek Czyzowicz[1],
Arnaud Labourel[2], Andrzej Pelc[1,*], Yann Vaxès[2]
[1]Université du Québec en Outaouais, C.P. 1250, succ. Hull, Gatineau, Qc. J8X 3X7 Canada.
E-mails: ingjuliananaya@gmail.com, jurek@uqo.ca, pelc@uqo.ca
[2] LIF, CNRS & Aix-Marseille University, 13453 Marseille, France.
E-mails: {jeremie.chalopin,arnaud.labourel,yann.vaxes}@lif.univ-mrs.fr

Abstract

A set of identical, mobile agents is deployed in a weighted network. Each agent possesses a battery - a power source allowing to move along network edges. Agent uses its battery proportionally to the distance traveled. At the beginning, each agent has its initial information. The agents exchange the actually possessed information when they meet. The agents collaborate in order to perform an efficient *convergecast*, where the initial information of all agents must be eventually transmitted to some agent.

The objective of this paper is to investigate what is the minimal value of power, initially available to all agents, so that convergecast may be achieved. We study the question in the centralized and the distributed setting. In the distributed setting every agent has to perform an algorithm being unaware of the network. We give a linear-time centralized algorithm solving the problem for line networks. We give a 2-competitive distributed algorithm achieving convergecast. The competitive ratio of 2 is proved to be the best possible for this problem, even if we only consider line networks. We show that already for the case of tree networks the centralized problem is strongly NP-complete. We give a 2-approximation centralized algorithm for general graphs.

1 Introduction

The model and the problem

A set of agents is deployed in a network represented by a weighted graph G. An edge weight represents its length, i.e., the distance between its endpoints along the edge. The agents start at different nodes of G. Every agent has a battery : a power source allowing it to move in a continuous way along the network edges. An agent may stop at any point of a network edge (i.e. at any distance from the edge endpoints, up to the edge weight). Agent's movements imply using its battery proportionally to the distance traveled. We assume that all agents move at the same speed that is equal to one, i.e., the values of the distance traveled and the time spend while travelling are commensurable. At the start of the algorithm, the agents start with the same amount of power noted P, allowing all agents to travel the same distance P.

Initially, each agent has an individual piece of information. When two (or more) agents are at the same point of the network at the same time, they automatically detect each other's presence and they exchange their information, i.e., each agent transmits all its possessed information to all other agents present at the point (hence an agent transmits information collected during all previous meetings). The purpose of a *convergecast* algorithm is to schedule the movements of the agents, so that the exchanges of the currently possessed information between the meeting agents eventually result in some agent, not a priori predetermined, containing the union of individual information of all the agents. This task is important, e.g., when

*Partially supported by NSERC discovery grant and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

agents have partial information about the topology of the network and the aggregate information can be used to construct a map of it, or when individual agents hold measurements performed by sensors located at their initial positions and collected information serves to make some global decision based on all measurements.

Agents try to cooperate so that the convergecast is achieved with the smallest possible agent's initial battery power P_{OPT}, i.e., minimizing the maximum distance traveled by an agent. We investigate the problem in two possible settings, centralized and distributed.

In the centralized setting, the problem must be solved by a centralized authority knowing the network and the initial positions of all the agents. We call *strategy* a finite sequence of movements executed by the agents. During each movement, starting at a specific time, an agent walks between two points belonging to the same network edge. A strategy is a convergecast strategy if the sequence of movements results in one agent possessing the initial information of every agent. We consider two different versions of the problem : the decision problem, i.e., deciding if there exists a convergecast strategy using power P (where P is the input of the problem) and the optimization problem, i.e., computing the smallest amount of power that is sufficient to achieve convergecast.

In the distributed question, the problem must be approached individually by each agent. Each agent is unaware of the network, of its position in the network and without the knowledge of positions (or even the presence) of any other agents. The agents are anonymous, i.e., they must execute the same algorithm. Each agent has a very simple sensing device allowing it to detect the presence of other agents at its location in the network. The agent is also aware of the degree of the node at which it is located as well as the port through which it enters a node, called *entry port*. We assume that the ports of a d-degree node are represented by integers $1, 2, \ldots d$. When two or more agents meet, each of them is aware of the direction from which the other agent is coming, i.e., the last entry port of each agent. Each agent has memory sufficient to store all information initially belonging to all agents as well as a small (constant) number of real values. Since the measure of efficiency in this paper is the battery power (or the maximum distance traveled by an agent, which is proportional to the battery power used) we do not try to optimize the other resources (e.g. global execution time, local computation time, memory size of the agents, communication bandwidth, etc.). In particular, we conservatively suppose that, whenever two agents meet, they automatically exchange the entire information they possess (rather than the new information only). This information exchange procedure is never explicitly mentioned in our algorithms, supposing, by default, that it always takes place when a meeting occurs. The efficiency of a distributed solution is expressed by the *competitive ratio*, which is the worst-case ratio of the amount of power necessary to solve the convergecast by the distributed algorithm with respect to the amount of power computed by the optimal centralized algorithm, which is executed for the same agents' initial positions.

It is easy to see, that in the optimal centralized solution for the case of the line and the tree, the original network may be truncated by removing some portions and leaving only the connected part of it containing all the agents (this way all leaves of the remaining tree contain initial positions of agents). We make this assumption also in the distributed setting, since no finite competitive ratio is achievable if this condition is dropped. Indeed, two nearby anonymous agents inside a long line need to travel a long distance to one of its endpoints to break symmetry in order to meet.

Related work

Rapidly developing network and computer industry fueled the research interest in mobile agents (robots) computing. Mobile agents are often interpreted as software agents, i.e., programs migrating from host to host in a network, performing some specific tasks. However, the recent developments in computer technology bring up specific problems related to physical mobile devices. These include robots or motor vehicles, various wireless gadgets, or even living mobile agents: humans (e.g. soldiers on the battlefield or emergency disaster relief personnel) or animals (e.g. birds, swarms of insects).

In many applications the involved mobile agents are small and have to be produced at low cost in massive

numbers. Consequently, in many papers, the computational power of mobile agents is assumed to be very limited and feasibility of some important distributed tasks for such collections of agents is investigated. For example [6] introduced *population protocols*, modeling wireless sensor networks by extremely limited finite-state computational devices. The agents of population protocols move according to some mobility pattern totally out of their control and they interact randomly in pairs. This is called *passive mobility*, intended to model, e.g., some unstable environment, like a flow of water, chemical solution, human blood, wind or unpredictable mobility of agents' carriers (e.g. vehicles or flocks of birds). On the other hand, [37] introduced anonymous, oblivious, asynchronous, mobile agents which cannot directly communicate, but they can occasionally observe the environment. Gathering and convergence [5, 19, 20, 21], as well as pattern formation [23, 25, 37, 38] were studied for such agents.

Apart from the feasibility questions for such limited agents, the optimization problems related to the efficient usage of agents' resources have been also investigated. Energy management of (not necessarily mobile) computational devices has been a major concern in recent research papers (cf. [1]). Fundamental techniques proposed to reduce power consumption of computer systems include power-down strategies (see [1, 8, 30]) and speed scaling (introduced in [39]). Several papers proposed centralized [17, 36, 39] or distributed [1, 4, 8, 30] algorithms. However, most of this research on power efficiency concerned optimization of overall power used. Similar to our setting, assignment of charges to the system components in order to minimize the maximal charge has a flavor of another important optimization problem which is load balancing (cf. [10]).

In wireless sensor and ad hoc networks the power awareness has been often related to the data communication via efficient routing protocols (e.g. [4, 36]. However in many applications of mobile agents (e.g. those involving actively mobile, physical agents) the agent's energy is mostly used for it's mobility purpose rather than communication, since active moving often requires running some mechanical components, while communication mostly involves (less energy-prone) electronic devices. Consequently, in most tasks involving moving agents, like exploration, searching or pattern formation, the distance traveled is the main optimization criterion (cf. [2, 3, 11, 12, 15, 16, 22, 24, 26, 33]). Single agent exploration of an unknown environment has been studied for graphs, e.g. [2, 22], or geometric terrains, [12, 16].

While a single agent cannot explore an unknown graph unless pebble (landmark) usage is permitted (see [13]), a pair of robots is able to explore and map a directed graph of maximal degree d in $O(d^2n^5)$ time with high probability (cf. [14]). In the case of a team of collaborating mobile agents, the challenge is to balance the workload among the agents so that the time to achieve the required goal is minimized. However this task is often hard (cf. [28]), even in the case of two agents on a tree, [9]. On the other hand, [26] study the problem of agents exploring a tree showing $O(k/\log k)$ competitive ratio of their distributed algorithm provided that writing (and reading) at tree nodes is permitted.

Assumptions similar to our paper have been made in [11, 16, 24] where the mobile agents are constrained to travel a fixed distance to explore an unknown graph, [11, 16], or tree, [24]. In [11, 16] a mobile agent has to return to its home base to refuel (or recharge its battery) so that the same maximal distance may repeatedly be traversed. [24] gives an 8-competitive distributed algorithm for a set of agents with the same amount of power exploring the tree starting at the same node.

The convergecast problem is sometimes viewed as a special case of the data aggregation question (e.g. [32, 35]) and it has been studied mainly for wireless and sensor networks, where the battery power usage is an important issue (cf. [31, 7]). Recently [18] considered the online and offline settings of the scheduling problem when data has to be delivered to mobile clients while they travel within the communication range of wireless stations. [31] presents a randomized distributed convergecast algorithm for geometric ad-hoc networks and study the trade-off between the energy used and the latency of convergecast. To the best of our knowledge, the problem of the present paper, when the mobile agents perform convergecast, by exchanging the possessed information when meeting, while optimizing the maximal power used by a mobile agent, has never been investigated.

Our results

In the case of centralized setting we give a linear-time deterministic algorithm finding an optimal convergecast strategy for line networks. We show that, already for the case of tree networks, the centralized problem is strongly NP-complete. We give a 2-approximation centralized algorithm for general graphs.

For the distributed setting, we show that the convergecast is possible for tree networks if all agents have the amount of initial power equal to twice the power necessary to achieve centralized convergecast. The competitive ratio of 2 is proved to be the best possible for this problem, even if we only consider line networks.

2 Centralized convergecast on lines

In this section we consider the centralized convergecast problem for lines. We give an optimal, linear-time, deterministic centralized algorithm, computing the optimal amount of power needed to solve convergecast for line networks. As the algorithm is quite involved, we start by observing some properties of the optimal strategies. Already relatively apparent properties permit to design an intuitive decision procedure, verifying whether a given amount of power is sufficient to perform convergecast. Then we present other ingredients needed for the linear-time optimization procedure.

We order agents according to their positions on the line. Hence we can assume w.l.o.g., that agent a_i, for $1 \leq i \leq n$ is initially positioned at point $Pos[i]$ of the line of length ℓ and that $Pos[1] = 0 < Pos[2] < \ldots < Pos[n] = \ell$.

2.1 Properties of a convergecast strategy

In this subsection, we show that if we are given a convergecast strategy for some configuration, then we can always modify it in order to get another convergecast strategy, using the same amount of maximal power for every agent, satisfying some interesting properties. These observations permit to restrict the search for the optimal strategy to some smaller and easier to handle subclass of strategies.

Observe that, in order to aggregate the entire information at a single point of the line, every agent a_i, for $1 < i < n$, must learn either the initial information of agent a_1 or a_n. Therefore, we can partition the set of agents performing a convergecast strategy into two subsets LR and RL, such that each agent $a_i \in LR$ learns the initial information of agent a_1 before learning the initial information of agent a_n (or not learning at all the information of a_n). All other agents belong to RL. For any convergecast strategy all the points visited by agent a_i form a real interval containing its initial position $Pos[i]$. We denote by $[b_i, f_i]$ the interval of all points visited by $a_i \in LR$ and by $[f_j, b_j]$ - the points visited by $a_j \in RL$.

In the next lemma, we show a necessary and sufficient condition for the existence of a convergecast strategy. It also shows that any convergecast strategy may be converted to a strategy that we call *regular* having particular properties. Firstly, each agent from LR of a regular strategy is initially positioned left to all agents of RL. Secondly, each agent of regular strategy needs to change its direction at most once. More precisely, each agent $a_i \in LR$ first goes back to a point $b_i \leq Pos[i]$, getting there the information from the previous agent (except a_1 that has no information to collect), then it goes forward to a point $f_i \geq b_i$. Similarly, each agent in RL first goes back to a point $b_i \geq Pos[i]$ and then moves forward to a point $f_i \leq b_i$. Moreover, we assume that each agent of a regular strategy travels the maximal possible distance, i.e., it spends all its power.

Lemma 1 *There exists a convergecast strategy $\mathcal{S}$ for a configuration $Pos[1:n]$ if and only if there exists a partition of the agents into two sets LR and RL and if for each agent a_i, there exist two points b_i, f_i of segment $[0, \ell]$ such that*

1. *there exists p such that $LR = \{a_i \mid i \leq p\}$ and $RL = \{a_i \mid i > p\}$,*
2. *if $a_i \in LR$, $b_i = \min\{f_{i-1}, Pos[i]\}$ and $f_i = 2b_i + P - Pos[i]$,*

3. *if* $a_i \in RL$, $b_i = \max\{f_{i+1}, Pos[i]\}$ *and* $f_i = 2b_i - P - Pos[i]$,
4. $\max\{f_i \mid a_i \in LR\} \geq \min\{f_i \mid a_i \in RL\}$.

In the following, we only consider regular strategies. Note that a regular strategy is fully determined by the value of P and by the partition of the agents into the two sets LR and RL. For each agent $a_i \in LR$ (resp. $a_i \in RL$), we denote f_i by $Reach_{LR}(i, P)$ (resp. $Reach_{RL}(i, P)$). Observe that $Reach_{LR}(i, P)$ is the rightmost point on the line to which the set of i agents at initial positions $Pos[1 : i]$, each having power P, may transport the union of their initial information. Similarly, $Reach_{RL}(i, P)$ is the leftmost such point for agents at positions $Pos[i : n]$.

Lemma 1 permits to construct a linear-time decision procedure verifying if a given amount P of battery power is sufficient to design a convergecast strategy for a given configuration $Pos[1 : n]$ of agents. We first compute two lists $Reach_{LR}(i, P)$, for $1 \leq i \leq n$ and $Reach_{RL}(i, P)$, for $1 \leq i \leq n$. Then we scan them to determine if there exists an index j, such that $Reach_{LR}(j, P) \geq Reach_{RL}(j + 1, P)$. In such a case, we set $LR = \{a_r \mid r \leq j\}$ and $RL = \{a_r \mid r > j\}$ and we apply Lemma 1 to obtain a convergecast strategy where agents a_j and a_{j+1} meet and exchange their information which totals to the entire initial information of the set of agents. If there is no such index j, no convergecast strategy is possible. This implies

Corollary 1 *In $O(n)$ time we can decide if a configuration of n agents on the line, each having a given maximal power P, can perform convergecast.*

The remaining lemmas of this subsection bring up observations needed to construct an $O(n)$ algorithm designing an optimal convergecast strategy.

Note that if the agents are not given enough power, then it can happen that some agent a_p may never learn the information from a_1 (resp. from a_n). In this case, a_p cannot belong to LR (resp. RL). We denote by $Act_{LR}(p)$ the minimum amount of power we have to give the agents to ensure that a_p can learn the information from a_1: if $p > 0$, $Act_{LR}(p) = \min\{P \mid Reach_{LR}(p - 1, P) + P \geq Pos[p]\}$. Similarly, we have $Act_{RL}(p) = \min\{P \mid Reach_{RL}(p + 1, P) - P \leq Pos[p]\}$.

Given a strategy using power P, for each agent $p \in LR$, we have $P \geq Act_{LR}(p)$ and either $Reach_{LR}(p - 1, P) \geq Pos[p]$, or $Reach_{LR}(p - 1, P) \leq Pos[p]$. In the first case, $Reach_{LR}(p, P) = Pos[p] + P$, while in the second case, $Reach_{LR}(p, P) = 2Reach_{LR}(p - 1, P) + P - Pos[p]$.

We define threshold functions $TH_{LR}(p)$ and $TH_{RL}(p)$ that compute for each index p, the minimal amount of agents' power ensuring that agent a_p does not go back when $a_p \in LR$ or $a_p \in RL$ respectively (i.e. such that $b_p = Pos[p]$). For each p, let $TH_{LR}(p) = \min\{P \mid Reach_{LR}(p, P) = Pos[p] + P\}$ and $TH_{RL}(p) = \min\{P \mid Reach_{RL}(p, P) = Pos[p] - P\}$. Clearly, $TH_{LR}(1) = TH_{RL}(n) = 0$.

The next lemma illustrates how to compute $Reach_{LR}(q, P)$ and $Reach_{RL}(q, P)$ if we know $TH_{LR}(p)$ and $TH_{RL}(p)$ for every agent p.

Lemma 2 *Consider an amount of power P and an index q. If $p = \max\{p' \leq q \mid TH_{LR}(p') < P\}$, then $Reach_{LR}(q, P) = 2^{q-p}Pos[p] + (2^{q-p+1} - 1)P - \sum_{i=p+1}^{q} 2^{q-i}Pos[i]$. Similarly, if $p = \min\{p' \geq q \mid TH_{RL}(p') < P\}$, then $Reach_{RL}(q, P) = 2^{p-q}Pos[p] - (2^{p-q+1} - 1)P - \sum_{i=q}^{p-1} 2^{i-q}Pos[i]$.*

Observe that the previous lemma implies that, for each q, the function $Reach_{LR}(q, \cdot)$ is an increasing, continuous, piecewise linear function on $[Act_{LR}(q), +\infty)$ and that $Reach_{RL}(q, \cdot)$ is a decreasing, continuous, piecewise linear function on $[Act_{RL}(q), +\infty)$.

In the following, we denote $S_{LR}(p, q) = \sum_{i=p+1}^{q} 2^{q-i}Pos[i]$ and $S_{RL}(p, q) = \sum_{i=q}^{p-1} 2^{i-q}Pos[i]$.

Remark 2.1 *For every $p \leq q \leq r$, $S_{LR}(p, r) = 2^{r-q}S_{LR}(p, q) + S_{LR}(q, r)$.*

We now show that for an optimal convergecast strategy, the last agent of LR and the first agent of RL meet at some point between their initial positions and that they need to use all the available power to meet.

Lemma 3 *Suppose there exists an optimal convergecast strategy for a configuration $Pos[1:n]$, where the maximum power used by an agent is P. Then, there exists an integer $1 \leq p < n$ such that $Pos[p] < Reach_{LR}(p,P) = Reach_{RL}(p+1,P) < Pos[p+1]$.*

Moreover, $\forall q \leq p$, $Act_{LR}(q) < P < TH_{RL}(q)$ and $\forall q > p$, $Act_{RL}(q) < P < TH_{LR}(q)$.

2.2 A linear algorithm to compute the optimal power needed for convergecast

In this section, we prove the following theorem.

Theorem 1 *In $O(n)$ time, one can compute the optimal power needed to achieve convergecast on the line.*

We first explain how to compute a stack of couples $(p, TH_{LR}(p))$ that we can subsequently use to calculate $Reach_{LR}(p,P)$ for any given P. Then, we present a linear algorithm that computes the value needed to solve convergecast when the last index $r \in LR$ is provided: given an index r, we compute the optimal power needed to solve convergecast assuming that $LR = \{a_q \mid q \leq r\}$ and $RL = \{a_q \mid q > r\}$. Finally, we explain how to use techniques introduced for the two previous algorithms in order to compute the optimal power needed to solve convergecast.

Computing the thresholds values. To describe explicitly the function $Reach_{LR}(q,\cdot)$, we need to identify the indexes p such that for every $r \in [p+1,q]$, we have $TH_{LR}(r) > TH_{LR}(p)$. They correspond to the breakpoints at which the slopes of the piecewise linear function $Reach_{LR}(q,\cdot)$ change. Indeed, if we are given such an index p, then for every P comprised between $TH_{LR}(p)$ and $\min\{TH_{LR}(r) \mid p < r \leq q\}$, we have $Reach_{LR}(q,P) = 2^{q-p}Pos[p] + (2^{q-p+1}-1)P - S_{LR}(p,q)$. We denote by $X_{LR}(q)$ this set of indexes $\{p \leq q \mid \forall r \in [p+1,q], TH_{LR}(r) > TH_{LR}(p)\}$.

In particular, if we want to compute $TH_{LR}(q+1)$, we just need to find $p = \max\{r \leq q \mid Reach_{LR}(q, TH_{LR}(r)) < Pos[q+1]\}$, and then $TH_{LR}(q+1)$ is the value of power P such that $2^{q-p}Pos[p] + (2^{q-p+1}-1)P - S_{LR}(p,q) = Pos[q+1]$. Moreover, by the choice of p, we have $X_{LR}(q+1) = \{r \in X_{LR}(q) \mid r \leq p\} \cup \{q+1\}$.

Using these remarks, the function `ThresholdLR`, having given an agent index r, returns a stack $\texttt{TH}_{\texttt{LR}}$ containing couples (p,P) such that $p \in X_{LR}(r)$ and $P = TH_{LR}(p)$. Note that in the stack $\texttt{TH}_{\texttt{LR}}$, the elements (p,P) are sorted along both components, the largest being on the top of the stack.

The algorithm proceeds as follows. Initially, the stack $\texttt{TH}_{\texttt{LR}}$ contains only the couple $(1, TH_{LR}(1))$. At each iteration, given the stack corresponding to the index q, in order to compute the stack for the index $q+1$, we first pop out all elements (p,P) such that $Reach_{LR}(q,P) > Pos[q+1]$. After that, the integer p needed to compute $TH_{LR}(q+1)$ is located on the top of the stack. Finally, the couple $(q+1, TH_{LR}(q+1))$ is pushed on the stack before we proceed with the subsequent index q. At the end of the procedure, we return the stack $\texttt{TH}_{\texttt{LR}}$ corresponding to the index r.

The number of stack operations performed during the execution of this function is $O(r)$. However, in order to obtain a linear number of arithmetic operations, we need to be able to compute 2^{q-p} and $S_{LR}(p,q)$ in constant time.

In order to compute 2^{q-p} efficiently, we can store the values of 2^i, $i \in [1, n-1]$ in an auxiliary array, that we have precomputed in $O(n)$ time. We cannot precompute all values of $S_{LR}(p,q)$ since this requires calculating $\Theta(n^2)$ values. However, from Remark 2.1, we know that $S_{LR}(p,q) = S_{LR}(1,q) - 2^{q-p}S_{LR}(1,p)$. Consequently, it is enough to precompute $S_{LR}(1,i)$ for each $i \in [2,n]$. Since $S_{LR}(1,i+1) = 2S_{LR}(1,i) + Pos[i+1]$, this can be done using $O(n)$ arithmetic operations.

Function `ThresholdLR(array` $Pos[1:n]$ `of real; r:integer):stack`

$\texttt{TH}_{\texttt{LR}} =$ **empty_stack;**
push $(\texttt{TH}_{\texttt{LR}}, (1,0))$;
for $q = 1$ **to** $r-1$ **do**
  $(p,P) =$ **pop**$(\texttt{TH}_{\texttt{LR}})$; /* $p = q$ and $P = TH_{LR}(p)$ */
  while $2^{q-p} * Pos[p] + (2^{q-p+1}-1) * P - S_{LR}(p,q) \geq Pos[q+1]$ **do** $(p,P) =$ **pop**$(\texttt{TH}_{\texttt{LR}})$;
  /* while $Reach_{LR}(q,P) \geq Pos[q+1]$ we consider the next element in $\texttt{TH}_{\texttt{LR}}$ */
  push $(\texttt{TH}_{\texttt{LR}}, (p,P))$;
  $Q = (2^{q-p} * Pos[p] - Pos[q+1] - S_{LR}(p,q))/(2^{q-p+1}-1)$;
  /* Q is the solution of $Reach_{LR}(q,P) = Pos[q+1]$ */
  push $(\texttt{TH}_{\texttt{LR}}, (q+1,Q))$;
return $(\texttt{TH}_{\texttt{LR}})$;

Similarly, we can define the function `ThresholdRL (array` $Pos[1:n]$ `of real, r:integer):stack` that returns a stack $\texttt{TH}_{\texttt{RL}}$ containing all pairs $(q, TH_{RL}(q))$ such that for every $p \in [r, q-1]$, we have $TH_{RL}(p) > TH_{RL}(q)$.

Computing the optimal power when LR and RL are known. Suppose now that we are given an agent index r and we want to compute the optimal power needed to solve convergecast when $LR = \{a_p \mid p \leq r\}$ and $RL = \{a_q \mid q > r\}$. From Lemma 3, we know that there exists a unique P_{OPT} such that $Reach_{LR}(r, P_{OPT}) = Reach_{RL}(r+1, P_{OPT})$.

As previously, by Lemma 2, we know that the value of $Reach_{LR}(r, P_{OPT})$ depends on $p = \max\{p' \leq r \mid TH_{LR}(p') < P_{OPT}\}$. Similarly, $Reach_{RL}(r+1, P_{OPT})$ depends on $q = \max\{q' \geq r+1 \mid TH_{RL}(q') < P_{OPT}\}$. If we are given the values of p and q, then P_{OPT} is the value of P such that

$$2^{r-p}Pos[p] - (2^{r-p+1}-1)P - S_{LR}(p,r) = 2^{q-r-1}Pos[q] - (2^{q-r}-1)P - S_{RL}(q,r+1).$$

In Algorithm `OptimalAtIndex`, we first use the previous algorithm to compute the two stacks $\texttt{TH}_{\texttt{LR}}$ and $\texttt{TH}_{\texttt{RL}}$ containing respectively $\{(p, TH_{LR}(p)) \mid p \in X_{LR}(r)\}$ and $\{(q, TH_{LR}(q)) \mid q \in X_{RL}(r+1)\}$. Then at each iteration, we consider the two elements (p, P_{LR}) and (q, P_{RL}) that are on top of both stacks. If $P_{LR} \geq P_{RL}$ (the other case is symmetric), we check whether $Reach_{LR}(r, P_{LR}) \geq Reach_{RL}(r+1, P_{LR})$. In this case, we have $P > P_{OPT}$, so we remove (p, P_{LR}) from the stack $\texttt{TH}_{\texttt{LR}}$ and we proceed to the next iteration. If $Reach_{LR}(r, P_{LR}) < Reach_{RL}(r+1, P_{LR})$, we know that $P_{OPT} \geq P_{LR} \geq P_{RL}$ and we can compute the value of P_{OPT} using Lemma 2.

Function `OptimalAtIndex(array` $Pos[1:n]$ `of real; r:integer):stack`

$\texttt{TH}_{\texttt{LR}} =$ `ThresholdLR`(r); $\texttt{TH}_{\texttt{RL}} =$ `ThresholdRL`$(r+1)$;
$(p, P_{LR}) =$ **pop**$(\texttt{TH}_{\texttt{LR}})$; $(q, P_{RL}) =$ **pop**$(\texttt{TH}_{\texttt{RL}})$; $P = \max\{P_{LR}, P_{RL}\}$;
/* $p = r$, $P_{LR} = TH_{LR}(r)$, $q = r+1$, $P_{RL} = TH_{RL}(r+1)$. */
while $2^{r-p}Pos[p] + (2^{r-p+1}-1)P - S_{LR}(p,r) \geq 2^{q-r-1}Pos[q] - (2^{q-r}-1)P - S_{RL}(q,r+1)$ **do**
/* While $Reach_{LR}(r,P) \geq Reach_{RL}(r+1,P)$ do */
  if $P_{LR} \geq P_{RL}$ **then** $(p, P_{LR}) =$ **pop**$(\texttt{TH}_{\texttt{LR}})$;
  else $(q, P_{RL}) =$ **pop**$(\texttt{TH}_{\texttt{RL}})$;
  $P = \max\{P_{LR}, P_{RL}\}$;
$P_{OPT} = (2^{q-r-1}Pos[q] - S_{RL}(q,r+1) - 2^{r-p}Pos[p] + S_{LR}(p,r))/(2^{r-p+1} + 2^{q-r} - 2)$;
/* P_{OPT} is the solution of $Reach_{LR}(r, P_{OPT}) = Reach_{RL}(r+1, P_{OPT})$ */
return (P_{OPT});

Let $Y_{LR}(r, P)$ denote $\{(p, TH_{LR}(p)) \mid p \in X_{LR}(r)$ and $TH_{LR}(p) < P\}$ and $Y_{RL}(r+1, P) = \{(q, TH_{RL}(q)) \mid q \in X_{RL}(r+1)$ and $TH_{RL}(q) < P\}$.

Remark 2.2 *At the end of the execution of the function* `OptimalAtIndex`*,* $\texttt{TH}_{\texttt{LR}}$ *and* $\texttt{TH}_{\texttt{RL}}$ *contain respectively* $Y_{LR}(r, P_{OPT})$ *and* $Y_{RL}(r+1, P_{OPT})$.

Moreover, if initially the two stacks $\texttt{TH}_{\texttt{LR}}$ *and* $\texttt{TH}_{\texttt{RL}}$ *contain respectively* $Y_{LR}(r, P)$ *and* $Y_{RL}(r+1, P)$ *for some* $P \geq P_{OPT}$*, then the value computed by the algorithm is also* P_{OPT} *.*

Computing the optimal power for convergecast. We now explain how to compute the optimal amount of power needed to achieve convergecast using a linear number of operations. The pseudo-code of the of Algorithm `ComputeOptimal` is given in Appendix B.

Let $P_{<r}$ be the optimal value needed to solve convergecast when $\max\{s \mid a_s \in LR\} < r$, i.e., when the two agents whose meeting results in merging the entire information are a_i and a_{i+1} for some $i < r$. If $Reach_{LR}(r, P_{<r}) \leq Reach_{RL}(r+1, P_{<r})$, then $P_{<r+1} = P_{<r}$. However, if $Reach_{LR}(r, P_{<r}) > Reach_{RL}(r+1, P_{<r})$, then $P_{<r+1} < P_{<r}$ and $P_{<r+1}$ is the unique value of P such that $Reach_{LR}(r, P) = Reach_{RL}(r+1, P)$. This corresponds to the value returned by `OptimalAtIndex` (Pos, r).

The general idea of Algorithm `ComputeOptimal` is to iteratively compute the value of $P_{<r}$. If we need a linear time algorithm, we cannot call repeatedly the function `OptimalAtIndex`. However, from Remark 2.2, in order to compute $P_{<r+1}$ when $P_{<r+1} \leq P_{<r}$, it is enough to know $Y_{LR}(r, P_{<r})$ and $Y_{RL}(r+1, P_{<r})$. If we know $Y_{LR}(r, P_{<r})$ and $Y_{RL}(r+1, P_{<r})$, then we can use the same algorithm as in `Optimal-AtIndex` in order to compute $P_{<r+1}$. Moreover, from Remark 2.2, we also get $Y_{LR}(r, P_{<r+1})$ and $Y_{RL}(r+1, P_{<r+1})$ when we compute $P_{<r+1}$.

Before proceeding to the next iteration, we need to compute $Y_{LR}(r+1, P_{<r+1})$ and $Y_{RL}(r+2, P_{<r+1})$ from $Y_{LR}(r, P_{<r+1})$ and $Y_{RL}(r+1, P_{<r+1})$. Note that if $TH_{LR}(r) > P_{<r+1}$, then $Y_{LR}(r+1, P_{<r+1}) = Y_{LR}(r, P_{<r+1})$. If $TH_{LR}(r) \leq P_{<r+1}$, we can use the same algorithm as in `ThresholdLR` to compute $Y_{LR}(r+1, P_{<r+1}) = \{(p, TH_{LR}(p)) \mid p \in X_{LR}(r)\}$ from $Y_{LR}(r, P_{<r+1})$. Consider now $Y_{RL}(r+2, P_{<r+1})$. If $TH_{RL}(r+1) > P_{<r+1}$, then $(r+1, TH_{RL}(r+1)) \notin Y_{RL}(r+1, P_{<r+1})$, and $Y_{RL}(r+2, P_{<r+1}) = Y_{RL}(r+1, P_{<r+1})$. If $TH_{RL}(r+1) \leq P_{<r+1}$, then either $Pos[r+1] - P_{<r+1} \geq Reach_{RL}(r+1, P_{<r+1})$ if $P_{<r+1} = P_{<r}$, or $Pos[r+1] - P_{<r+1} = Reach_{RL}(r+1, P_{<r+1}) = Reach_{LR}(r, P_{<r+1})$ if $P_{<r+1} < P_{<r}$. In both cases, it implies that $Act_{LR}(r+1) \geq P_{<r+1}$. Therefore, by Lemma 3, $P_{<i} = P_{<r+1}$ for every $i \geq r+1$ and we can return the value of $P_{<r+1}$.

In Algorithm `ComputeOptimal`, at each iteration, the stack $\texttt{TH}_{\texttt{LR}}$ contains $Y_{LR}(r, P_{<r})$ (except its top element) and the stack $\texttt{TH}_{\texttt{RL}}$ contains $Y_{RL}(r+1, P_{<r})$ (except its top element). Initially, $\texttt{TH}_{\texttt{LR}}$ is empty and $\texttt{TH}_{\texttt{RL}}$ contains $O(n)$ elements. In each iteration, at most one element is pushed into the stack $\texttt{TH}_{\texttt{LR}}$ and no element is pushed into the stack $\texttt{TH}_{\texttt{RL}}$. Consequently, the number of stack operations performed by the algorithm is linear.

3 Distributed convergecast on trees

A configuration of convergecast on graphs is a couple (G, A) where G is the weighted graph encoding the network and A is the set of the starting nodes of the agents. Let $D(G, A) = \max_{\emptyset \subsetneq X \subsetneq A}\{\min_{x \in X, y \in A \setminus X}\{d_G(x, y)\}\}$ where $d_G(x, y)$ is the distance between x and y in G. Clearly, we have $D(G, A) \leq 2P_{OPT}$.

We consider weighted trees with agents at every leaf. The next theorem states that there exists a 2-competitive distributed algorithm for the convergecast problem on trees.

Theorem 2 *Consider a configuration* (T, A) *where* T *is a tree and* A *contains all the leaves of* T*. There is a distributed convergecast algorithm using* $D(T, A) \leq 2P_{OPT}$ *power per agent.*

Sketch of the proof : In order to perform the convergecast, each agent executes Algorithm 1.

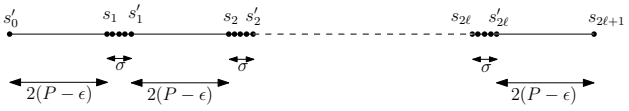

Figure 1: The configuration in the proof of Theorem 3.

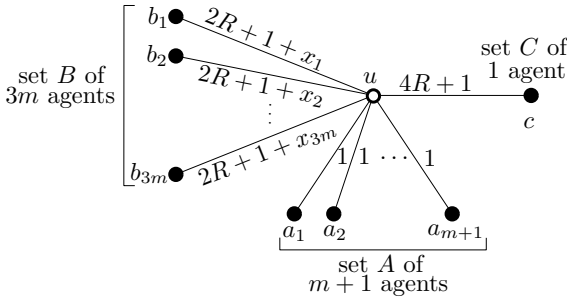

Figure 2: Instance of centralized convergecast problem from an instance of 3-partition in the proof of Theorem 4.

Algorithm 1: UnknownTree

```
collecting = true;
while collecting = true do
    Wait until there is at most one port unused by the agent incoming at the current node;
    if all ports of current node were used by incoming agents then collecting = false;
    if the agent has used less power than any other agent present at the node and collecting = true then
        Move through the unused incoming port until you meet another agent or reach a node;
    else collecting = false;
    if the agent is inside an edge then collecting = false;
```

The agents traverse the leaf edges then edges between nodes at height one and two and so on. When all the tree is traversed, all the information is collected at the last meeting point. No agent will use more power than $D(T, A) \leq 2P_{OPT}$. □

The following theorem shows that no distributed algorithm may offer a better competitive ratio than 2 even if we only consider line networks.

Theorem 3 *Consider any $\delta > 0$, and any value of power P. There exists an integer n and a configuration $Pos[1:n]$ of n agents on the line such that there is a convergecast strategy using power P and so that there is no deterministic distributed strategy allowing the agents to solve convergecast when the amount of power given to each agent is $(2-\delta)P$.*

Sketch of the proof : Let $\varepsilon = \delta P/4$ and $\sigma = \varepsilon/2 = \delta P/8$. Let $l = \lfloor \log(8/\delta) \rfloor$ and $k = l + 2$.

Consider a set of agents positioned on a line as follows (See Figure 1). There is an agent a_0 (resp. a_{2l+1}) on the left (resp. right) end of the line on position $s'_0 = 0$ (resp. s_l). For each $1 \leq i \leq 2l$, there is a set A_i of k agents on distinct initial positions within a segment $[s_i, s'_i]$ of length σ such that for each $1 \leq i \leq 2l+1$, the distance between s_i and s'_{i-1} is $2(P-\varepsilon)$. Using Lemma 2, we can show, that if the amount of power given to each agent is P, then convergecast is achievable.

Suppose now that there exists a distributed strategy $\mathcal{S}$ that solves convergecast on the configuration when the amount of power given to each agent is $(2-\delta)P$. We can show that for each $i \in [1, l]$, all agents from A_i perform the same moves as long as the leftmost agent of A_i has not met any agent from A_{i-1}. We show by induction on $i \in [1, l]$ that agents in A_i learn the information from a_0 before the one from a_{2l+1} and that no agent in the set A_i knowing the information from a_0 can reach the point $s_{i+1} - (2^{i+2} - 2)\varepsilon$. If we consider the agents from A_l, we get that no agent from A_{l-1} can reach $s_l - (2^{l+1} - 1)\varepsilon \leq s_l - (8/\delta - 2)\delta P/4 = s_l - 2P + \delta P/2 < s_l - 2P + \delta P$ having the initial information of a_0. Since no agent from the set A_l can reach any point on the left of $s_l - 2P + \delta P$, it implies that no agent from A_l can ever learn the information from a_0 and thus, $\mathcal{S}$ is not a distributed convergecast strategy. □

4 Centralized convergecast on trees and graphs

We show in this section that for trees the centralized convergecast problem is substantially harder.

Theorem 4 *The centralized convergecast decision problem is strongly NP-complete for trees.*

Sketch of the proof : We construct a polynomial-time many-one reduction from the 3-Partition problem which is strongly NP-Complete [27]. The 3-partition problem answers the following question: can a given multiset S of $3m$ positive integers x_i such that $R/4 < x_i < R/2$ be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ of size three such that for $1 \leq j \leq m$, $\sum_{x \in S_j} x = R$? The instance of the centralized convergecast problem constructed from an instance of 3-partition is the star depicted in Figure 2. There is an agent at each leaf and each agent is given power equal to $2R+1$. We can assume that in any convergecast strategy, agents a_i first move to the center u, each agent b_i moves at distance x_i from u and agent c moves at distance $2R$ from u. Agents a_i, for $1 \leq i \leq m$, must then collect the information from agents b_i and finally agent a_{m+1} must move to c in order to complete the convergecast. Since agents a_i, while reaching u have the remaining power of $2R$ and that collecting information from b_i and return to node u costs $2x_i$ power, the instance of convergecast has a solution if and only if the original instance of 3-partition has a solution.

It remains to show that the problem is in NP. Given a strategy $\mathcal{S}$, the certificate of the instance encodes in chronological order the positions of meetings in which at least one agent learns a new piece of information. There is a polynomial number of such meetings called useful meetings. We can show that the strategy $\mathcal{S}'$, in which agents move via shortest paths to their useful meetings, is a convergecast strategy and uses less power than $\mathcal{S}$. If a useful meeting occurs on an edge, the certificate encodes the name of a variable d_i that represents the exact position inside the edge. Checking that we can assign values to d_i, such that each agent moves a distance less than P in $\mathcal{S}'$, can be done in polynomial time using linear programming. □

Even if the exact centralized optimization problem is NP-complete, we can obtain a 2-approximation of the power needed to achieve centralized convergecast in graphs in polynomial time.

Theorem 5 *Consider a configuration (G, A). There is a polynomial algorithm computing a convergecast strategy using $D(G, A) \leq 2P_{OPT}$ power per agent.*

Sketch of the proof : The idea of the algorithm is to construct a total order u_i on the positions of agents in the following way. We put in set V an arbitrary agent's initial position u_0. Iteratively, we add to the set V of already treated agents' positions a new agent's position u_i, which is at the closest distance from V along some path P_i. Then agents move in the reverse order, starting with agent at u_{k-1}. Agent at u_i moves following the path P_i. The length of P_i is less than $D(G, A)$. When all agents have moved, agent u_1 possesses all the information. □

5 Conclusion and open problems

It is worth pursuing questions related to information transportation by mobile agents to other communication problems, for example broadcasting or gossiping. Only some of our techniques on convergecast extend to these settings (e.g. NP-hardness for trees).

The problem of a single *information transfer* by mobile agents between two stationary points of the network is also interesting. In particular, it is an open question whether this problem for tree networks is still NP-hard or if a polynomial-time algorithm is possible, since our reduction to 3-partition is no longer valid.

Other related questions may involve agents with unequal power, agents with non-zero visibility, labeled agents, unreliable agents or networks, etc.

References

[1] S. Albers: Energy-efficient algorithms. *Comm. ACM 53(5)*, (2010), pp. 86-96.

[2] S. Albers, M.R. Henzinger. Exploring unknown environments. *SIAM J. on Comput.*, 29 (4), pp.1164-1188.

[3] S. Alpern and S. Gal, The theory of search games and rendezvous. *Kluwer Academic Publ.*, 2002.

[4] Ambuhl, C. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In *Proc. of 32nd ICALP* (2005), pp. 1139-1150.

[5] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. on Robotics and Automation*, 15(5) (1999), pp. 818-828.

[6] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, R. Peralta, Computation in networks of passively mobile finite-state sensors, *Distributed Computing* (2006), pp. 235-253.

[7] V. Annamalai, S. K. S. Gupta, and L. Schwiebert, On Tree-Based Convergecasting in Wireless Sensor Networks, *Wireless Communications and Networking, IEEE*, vol. 3 (2003), pp. 1942 - 1947.

[8] J. Augustine, S. Irani, C. Swamy. Optimal powerdown strategies. *SIAM J. Comput. 37* (2008), pp. 1499-1516.

[9] I. Averbakh, O. Berman. A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree. *Discrete Applied Mathematics*, 68 (1996), pp. 17-32.

[10] Y. Azar. On-line load balancing. In: *A. Fiat and G.Woeginger, Online Algorithms: The State of the Art, Springer* LNCS 1442, (1998), pp. 178-195.

[11] B. Awerbuch, M. Betke, R. Rivest, M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152 (1999), pp. 155-172.

[12] R.A. Baeza Yates, J.C. Culberson, and G.J.E. Rawlins. Searching in the Plane. *Information and Computation*, 106(2), (1993), pp. 234-252.

[13] M. Bender, A. Fernandez, D. Ron, A. Sahai, S. Vadhan. The power of a pebble: exploring and mapping directed graphs. In *Proc. 30th STOC*, (1998), pp. 269-278.

[14] M. Bender, D. Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. In *Proc. 35th FOCS* (1994), pp. 75-85.

[15] M. Betke, R.L. Rivest, M. Singh. Piecemeal learning of an unknown environment. *Machine Learning*, 18(2/3), (1995), pp. 231-254.

[16] A. Blum, P. Raghavan, B. Schieber. Navigating in unfamiliar geometric terrain, *SIAM J. Comput.*, 26(1), (1997), pp. 110-137.

[17] Bunde, D. P. Power-aware scheduling for makespan and flow. *SPAA* (2006), pp. 190-196.

[18] F. Chen, M. P. Johnson, Y. Alayev, A. Bar-Noy, T. F. La Porta, Who, When, Where: Timeslot Assignment to Mobile Clients, *IEEE Transactions on Mobile Computing*, (2012), vol. 11, no. 1, pp. 73-85.

[19] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro. Solving the Robots Gathering Problem, In *Proc ICALP*, (2003), pp. 1181-1196.

[20] R. Cohen and D. Peleg. Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems. *SIAM J. on Comput.*, 34(6), (2005), pp. 1516-1528.

[21] A. Cord-Landwehr, B. Degener, M. Fischer, M. Hullmann, B. Kempkes, A. Klaas, P. Kling, S. Kurras, M. Martens, F. Meyer auf der Heide, C. Raupach, K. Swierkot, D. Warner, C. Weddemann, D. Wonisch: A New Approach for Analyzing Convergence Algorithms for Mobile Robots. *Proc. ICALP (2)*, (2011), pp. 650-661.

[22] X. Deng, C. H. Papadimitriou, Exploring an unknown graph. In *Proc. 31st FOCS*, (1990), vol I, pp. 355-361.

[23] S. Das, P. Flocchini, N. Santoro, M. Yamashita. On the Computational Power of Oblivious Robots: Forming a Series of Geometric Patterns, In *Proc. PODC*, (2010), pp. 267-276.

[24] M. Dynia, M. Korzeniowski, C. Schindelhauer. Power-aware collective tree exploration. In *Proc. of ARCS* (2006), pp. 341-351.

[25] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer. Gathering of asynchronous robots with limited visibility, *Th. Comp. Science*, 337, (2005), pp. 147-168.

[26] P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc. Collective tree exploration. In *Proc. LATIN*, (2004), pp. 141-151.

[27] M. R. Garey, and D. S. Johnson, Computers and Intractability; *A Guide to the Theory of NP-Completeness* (1979). pp. 96–105 and 224.

[28] G. Frederickson, M. Hecht, C. Kim. Approximation algorithms for some routing problems. *SIAM J. on Comput.*, 7 (1978), pp. 178-193.

[29] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, An Application-Specific Protocol Architecture for Wireless Microsensor Networks, *Transactions on wireless communication*, Vol. 1, No. 4, (2002), pp. 660-670.

[30] S. Irani, S.K. Shukla, R. Gupta. Algorithms for power savings. *ACM Trans. on Algorithms*, Vol. 3, No. 4, Article 41, (2007).

[31] A. Kesselman and D. R. Kowalski, Fast distributed algorithm for convergecast in ad hoc geometric radio networks, *Journal of Parallel and Distributed Computing*, Vol. 66, No. 4 (2006), pp. 578-585.

[32] L. Krishnamachari, D. Estrin, S. Wicker. The impact of data aggregation in wireless sensor networks. *ICDCS Workshops*, (2002), pp. 575-578.

[33] N. Megow, K. Mehlhorn, P. Schweitzer. Online Graph Exploration: New Results on Old and New Algorithms. *Proc. ICALP (2)*, (2011), pp. 478-489.

[34] S. Nikoletseas and P. G. Spirakis. Distributed Algorithms for Energy Efficient Routing and Tracking in Wireless Sensor Networks, *Algorithms*, 2, (2009), pp. 121-157.

[35] R. Rajagopalan, P. K. Varshney, Data-aggregation techniques in sensor networks: a survey *Communications Surveys and Tutorials, IEEE*, Vol. 8 , No. 4, (2006), pp. 48 - 63.

[36] Stojmenovic, I. and Lin, X. Power-Aware Localized Routing in Wireless Networks. *IEEE Trans. Parallel Distrib. Syst.* 12(11), (2001), pp. 1122-1133.

[37] I. Suzuki and M. Yamashita, Distributed Anonymous Mobile Robots: Formation of Geometric Patterns, *SIAM J. Comput.*, vol. 28(4), (1999), pp. 1347-1363.

[38] M. Yamashita, I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Th. Comp. Science*, 411(26-28), 2010, pp. 2433-2453.

[39] F.F. Yao, A.J. Demers, S. Shenker, A scheduling model for reduced CPU energy. In *Proc. of 36th FOCS* (1995), pp. 374-382.

Appendix

A Proofs of Section 2.1

Before proving Lemma 1, we need an additional lemma.

Lemma 4 *Suppose that there exists a convergecast strategy $\mathcal{S}$ for a configuration $Pos[1:n]$ using maximum power P. Then there exists a convergecast strategy for the same initial configuration of agents $a_1, \ldots, a_n$, which uses power P, with the following property*

- *if $a_i \in LR$, $b_i \leq Pos[i]$, $b_i \leq f_i$, and $Pos[i] + f_i - 2b_i \leq P$*
- *if $a_i \in RL$, $b_i \geq Pos[i]$, $b_i \geq f_i$, and $2b_i - Pos[i] - f_i \leq P$*
- *$\cup_{a_i \in LR}[b_i, f_i]$ and $\cup_{a_i \in RL}[f_i, b_i]$ are intervals,*
- *$\max\{f_i \mid a_i \in LR\} \geq \min\{f_i \mid a_i \in RL\}$.*

Proof : First, suppose we are given a partition of the agents into two disjoint sets of agents LR and RL and values b_i, f_i for each agent a_i satisfying the conditions of the claim. Consider the following strategy: first, every agent $a_i \in LR \cup RL$ moves to b_i; subsequently, every agent in LR moves to f_i once it learns the information from a_1; then, every agent in RL moves to f_i once it learns the information from a_n. Let a_k be an agent from LR such that f_k is maximum. Once a_k has moved to f_k, it knows the initial information of all the agents a_i such that $b_i \leq f_k$. If $f_k \geq \ell$, convergecast is achieved. Otherwise, since $f_k = \max\{f_i \mid a_i \in LR\} \geq \min\{f_i \mid a_i \in RL\}$, we know that there exists an agent $a_j \in RL$ such that $f_j \leq f_k < b_j$. When a_j reaches f_k it knows the initial information of all the agents such that $b_i \geq f_k$ and thus, a_j and a_k know the initial information of all agents.

Consider now a strategy $\mathcal{S}$ that solves the convergecast problem for n agents at initial positions $Pos[1:n]$ on a line. For any agent $a_i \in LR$ (resp. $a_i \in RL$), we denote by x_i the point where a_i learns the information of a_1 (resp. a_n) and by f_i the rightmost (resp. leftmost) point visited by a_i after that. Let b_i be the minimum (resp. maximum) of x_i and $Pos[i]$. Since a_i uses at most P power, we have $|Pos[i]-x_i|+|f_i-x_i| \leq P$. First suppose that $b_i = x_i$, we have $(Pos[i]-b_i)+(f_i-b_i) \leq P$ and thus $Pos[i]+f_i-2b_i \leq P$ if $a_i \in LR$, and $2b_i-Pos[i]-f_i \leq P$ if $a_i \in RL$. Now, suppose that $b_i = Pos[i]$, we have $Pos[i]+f_i-2b_i = f_i - Pos[i] \leq P$ if $a_i \in LR$ and $2b_i - Pos[i] - f_i = Pos[i] - f_i \leq P$ if $a_i \in RL$.

Since $\mathcal{S}$ is a convergecast strategy, there exists $a_i \in LR$ and $a_j \in RL$ such that $f_i \geq f_j$, and consequently, $\max\{f_i \mid a_i \in LR\} \geq \min\{f_i \mid a_i \in RL\}$. Clearly, if $a_i \in LR$, either $b_i = 0$, or there exists $a_j \in LR$ such that $b_j < b_i \leq f_j$. Consequently, $\cup_{a_i \in LR}[b_i, f_i]$ is an interval. For similar reasons, $\cup_{a_i \in RL}[b_i, f_i]$ is also an interval. □

Proof of Lemma 1 : By Lemma 4, we know that there exists a partition of the agents into two sets LR and RL and that for each agent a_i, there exist positions b_i, f_i satisfying the conditions of Lemma 4.

In the following claims, we show that we can modify the sets LR and RL, and the values of b_i and f_i for $1 \leq i \leq n$ in order to satisfy the conditions of Lemma 1.

Claim 1 *For each $a_i \in LR$, we can assume that $f_i = 2b_i + P - Pos[i]$ and for each $a_i \in RL$, we can assume that $f_i = 2b_i - P - Pos[i]$.*

We prove the claim for $a_i \in LR$; the other case is similar. Consider $a_i \in LR$ such that $f_i < 2b_i + P - Pos[i]$. We replace f_i by $f_i' = 2b_i + P - Pos[i]$. Since $f_i' > f_i$ and $Pos[i] + f_i' - 2b_i \leq P$, the conditions of Lemma 4 are still satisfied and we once again have a convergecast strategy.

Claim 2 *For each $a_i \in LR$, we can assume that either $b_i = \max\{f_j \mid b_j \leq b_i\}$ or $b_i = Pos[j]$. For each $a_i \in RL$, we can assume that either $b_i = \min\{f_j \mid b_j \geq b_i\}$ or $b_i = Pos[j]$.*

We prove the claim for LR; the other case is similar. If $i = 1$, then $b_i = Pos[i] = 0$, and the claim holds. For each $a_i \in LR, i > 1$, let $A_i = \{a_j \in LR \mid b_j < b_i \text{ or } b_j = b_i \text{ and } j < i\}$ and let $F_i = \max\{f_j \mid a_j \in A_i\}$. By Lemma 4, either $b_i = 0$ and $a_1 \in A_i$, or there exists an agent $a_j \in LR$ such that $b_j < b_i \leq f_j$. In both cases, $A_i \neq \emptyset$ and F_i is well-defined. We replace b_i by $b'_i = \min\{Pos[i], F_i\} \geq b_i$, and f_i by $f'_i = 2b'_i + P - Pos[i] \geq f_i$. Since $[b_i, f_i] \subseteq \cup_{a_j \in A_i}[b_j, f_j] \cup [b'_i, f'_i]$, $\cup_{a_j \in LR}[b_j, f_j]$ is an interval and by Lemma 4, we once again have a convergecast strategy.

Claim 3 *For each $a_i, a_j \in LR$ (resp. $a_i, a_j \in RL$), we can assume that $Pos[i] < Pos[j]$ implies $b_i \leq b_j$.*

We prove the claim for LR; the other case is similar. Suppose this is not the case. Let i be the smallest index such that $a_i \in LR$ and such that there exists $a_j \in LR$ with $Pos[i] < Pos[j]$ and $b_j < b_i$. Since $b_i > b_j \geq 0$, by Lemma 4, there exists $a_k \in LR$ such that $b_k < b_i \leq f_k$. If $k < i$, then due to the choice of i, $b_k \leq b_j$, and thus by Claim 2, $b_j \geq f_k \geq b_i$, a contradiction. Thus, there exists $a_j \in LR$ such that $j > i$ and $b_j < b_i \leq f_j$.

We exchange the roles of a_i and a_j: we let $b'_i = b_j$, $f'_i = 2b_j + P - Pos[i] \geq f_j$, $b'_j = \min\{Pos[j], f'_i\}$ and $f'_j = 2b'_j + P - Pos[j]$. Thus, $[b'_i, f'_i] \cup [b'_j, f'_j] = [b'_i, f'_j]$. Moreover, $b'_i = b_j$ and either $f'_j = Pos[j] + P > Pos[i] + P \geq f_i$, or $f'_j = 2f'_i + P - Pos[j] = 4b_j + 3P - 2Pos[i] - Pos[j]$. In the second case, since $f_j = 2b_j + P - Pos[j] \geq b_i$ and since $Pos[i] \leq Pos[j]$, we have $f'_j = 2b_j + P - Pos[j] + 2b_j + P - Pos[i] + P - Pos[i] \geq 2b_i + P - Pos[i] = f_i$. Consequently, we have $[b_j, f_i] \subseteq [b'_i, f'_j]$, and by Lemma 4, we once again have a convergecast strategy.

Claim 4 *If $a_i \in LR$ and $a_j \in RL$, we can assume that $j > i$.*

Suppose there exists an index i such that $a_i \in RL$ and $a_{i+1} \in LR$. Let $F_{RL} = \min\{f_j \mid a_j \in RL, j > i\}$ and $F_{LR} = \max\{f_j \mid a_j \in LR, j < i\}$; note that $b_i = \max\{F_{RL}, Pos[i]\}$ and $b_{i+1} = \min\{F_{LR}, Pos[i+1]\}$. We exchange the roles of a_i and a_{i+1}, i.e., we put $a_i \in LR$ and $a_{i+1} \in RL$. Let $b'_i = \min\{F_{LR}, Pos[i]\}$, $b'_{i+1} = \max\{F_{RL}, Pos[i+1]\}$, $f'_i = 2b'_i + P - Pos[i]$ and $f'_{i+1} = 2b'_{i+1} - P - Pos[i+1]$.

If $F_{RL} \leq Pos[i+1]$, then $f'_{i+1} = Pos[i+1] - P \leq b_{i+1} \leq F_{LR}$. If $F_{LR} \geq Pos[i]$, then $f'_i = Pos[i] + P \geq b_i \geq F_{RL}$. In both cases, by Lemma 4, we still have a convergecast strategy.

If $F_{RL} \geq Pos[i+1]$ and $F_{LR} \leq Pos[i]$, then $f'_i = 2F_{LR} + P - Pos[i] > 2F_{LR} + P - Pos[i+1] = f_{i+1}$, and $f'_{i+1} = 2F_{RL} - P - Pos[i+1] < 2F_{RL} - P - Pos[i] = f_i$. Consequently, by Lemma 4, we once again have a convergecast strategy. □

Before proving Lemma 2, we prove that for each p, the functions $Reach_{LR}(p, \cdot)$ and $Reach_{RL}(p, \cdot)$ are piecewise linear.

Lemma 5 *For every $1 \leq p \leq n$, the function $Reach_{LR}(p, \cdot) : P \to Reach_{LR}(p, P)$ is an increasing, continuous, piecewise linear function on $[Act_{LR}(p), +\infty)$.*

For every $1 \leq p \leq n$, the function $Reach_{RL}(p, \cdot) : P \to Reach_{RL}(p, P)$ is a decreasing continuous piecewise linear function on $[Act_{RL}(p), +\infty)$.

Proof : We prove the first statement of the lemma by induction on p. For $p = 1$, $Reach_{LR}(1, P) = Pos[1] + P$ and the claim holds. Suppose that $Reach_{LR}(p, \cdot)$ is a continuous piecewise linear function on $[Act_{LR}(p), +\infty)$ and consider $Reach_{LR}(p+1, \cdot)$.

First note that $Act_{LR}(p) < Act_{LR}(p+1)$. Since $Reach_{LR}(p, \cdot)$ is a continuous, increasing function, there exists a unique $P = Act_{LR}(p+1)$ such that $Reach_{LR}(p, P) + P = Pos[p+1]$ and for every

$P' > Act_{LR}(p+1)$, $Reach_{LR}(p, P') + P' > Pos[p+1]$. Consequently, $Reach_{LR}(p+1, \cdot)$ is well defined on $[Act_{LR}(p+1), +\infty)$.

Since $Reach_{LR}(p, \cdot)$ is a continuous, increasing function, there exists a unique $P = TH_{LR}(p+1)$ such that $Reach_{LR}(p, P) = Pos[p+1]$. If $Act_{LR}(p+1) \geq P \geq TH_{LR}(p+1)$, $Reach_{LR}(p+1, P) = 2Reach_{LR}(p, P) + P - Pos[p+1]$ and thus $Reach_{LR}(p+1, \cdot)$ is an increasing, continuous, piecewise linear function on $[Act_{LR}(p+1), TH_{LR}(p+1)]$. If $P \geq TH_{LR}(p+1)$, $Reach_{LR}(P) = Pos[p+1] + P$ and thus, $Reach_{LR}(p+1, \cdot)$ is an increasing, continuous, piecewise linear function on $[TH_{LR}(p+1), +\infty)$. Since $2Reach_{LR}(p, TH_{LR}(p+1)) + TH_{LR}(p+1) - Pos[p+1] = Pos[p+1] + TH_{LR}(p+1)$, $Reach_{LR}(p+1, \cdot)$ is an increasing, continuous, piecewise linear function on $[Act_{LR}(p+1), +\infty)$.

One can show the second statement of the lemma using similar arguments. □

Proof of Lemma 2 : We prove the first claim of the lemma; the other case is similar. We first show by induction on $q - p$ that if for every $i \in [p+1, q]$, $P \leq TH_{LR}(i)$, then

$$Reach_{LR}(q, P) = 2^{q-p} Reach_{LR}(p, P) + (2^{q-p} - 1)P - \sum_{i=p+1}^{q} 2^{q-i} Pos[i].$$

Note that since $P \leq TH_{LR}(q)$, $Reach_{LR}(q, P) = 2Reach_{LR}(q-1, P) + P - Pos[q]$. Thus if $q = p+1$, the claim holds. Suppose now that $q > p+1$. Since $q - 1 > p$, by induction hypothesis, we have

$$Reach_{LR}(q-1, P) = 2^{q-1-p} Reach_{LR}(p, P) + (2^{q-1-p} - 1)P - \sum_{i=p+1}^{q-1} 2^{q-1-i} Pos[i].$$

Consequently, we have

$$\begin{aligned} Reach_{LR}(q, P) &= 2Reach_{LR}(q-1, P) + P - Pos[q] \\ &= 2^{q-p} Reach_{LR}(p, P) + (2^{q-p} - 2)P - \sum_{i=p+1}^{q-1} 2^{q-i} Pos[i] + P - Pos[q]. \\ &= 2^{q-p} Reach_{LR}(p, P) + (2^{q-p} - 1)P - \sum_{i=p+1}^{q} 2^{q-i} Pos[i]. \end{aligned}$$

In order to prove the lemma, note that if $p = \max\{p' \leq q \mid TH_{LR}(p') < P\}$, then for each $p' \in [p+1, q]$, $TH_{LR}(p') \geq P$ and $Reach_{LR}(p, P) = Pos[p] + P$. Consequently,

$$Reach_{LR}(q, P) = 2^{q-p} Pos[p] + (2^{q-p+1} - 1)P - \sum_{i=p+1}^{q} 2^{q-i} Pos[i].$$

□

Proof of Lemma 3 : Suppose we are given p and consider the partition of the agents into $LR = \{a_q \mid q \leq p\}$ and $RL = \{a_q \mid q > p\}$. Consider a convergecast strategy respecting this partition and where the maximum amount of power P used by an agent is minimized. We first show that $Reach_{LR}(p, P) = Reach_{RL}(p+1, P)$.

Let $Q = \max\{Act_{LR}(p), Act_{RL}(p+1)\}$. Since $Reach_{LR}(p, \cdot)$ is an increasing continuous function on $[Act_{LR}(p), +\infty)$ and $Reach_{RL}(p+1, \cdot)$ is a decreasing continuous function on $[Act_{RL}(p+1), +\infty)$, $Reach_{LR}(p, \cdot) - Reach_{RL}(p+1, \cdot)$ is a continuous increasing function on $[Q, +\infty)$.

Consider the case where $Q = Act_{RL}(p+1) \geq Act_{LR}(p)$ (the other case is similar). Since $Reach_{RL}(p+1, Q) = Reach_{RL}(p+2, Q) = Pos[p+1] + Q$, $Reach_{LR}(p, Q) \leq Pos[p] + Q < Pos[p+1] + Q =$

$Reach_{RL}(p+1, Q)$ and thus, $Reach_{LR}(p, Q) - Reach_{RL}(p+1, Q) < 0$. By Lemma 1, there exists Q' such that $Reach_{LR}(p, Q') - Reach_{RL}(p+1, Q') \geq 0$. Consequently, there exists a unique $Q < P \leq Q'$ such that $Reach_{LR}(p, P) = Reach_{RL}(p+1, P)$.

Consider now an optimal convergecast strategy and let P be the maximum amount of power used by any agent. By Lemma 1 and according to what we have shown above, we know there exists p such that $Reach_{LR}(p, P) = Reach_{RL}(p+1, P)$.

Suppose that $Reach_{LR}(p, P) \leq Pos[p]$. In this case, we have $Reach_{RL}(p, P) = Pos[p] - P < Reach_{LR}(p-1, P)$ since $P > Act_{LR}(p)$. Consequently, according to what we have shown above, there exists $P' < P$ such that $Reach_{RL}(p, P') \leq Reach_{LR}(p-1, P')$ and P is not the optimal value needed to solve convergecast.

For similar reasons, if $Reach_{RL}(p+1, P) > Pos[p+1]$, P is not the optimal value needed to solve convergecast.

We now prove that for each $q \in [1, p]$, $Act_{LR}(q) < P$. This follows from the fact that for each $a_q \in LR$ such that $q > 1$, we have $Act_{LR}(q) > Act_{LR}(q-1)$. Consequently, for each $q \in [1, p-1]$, $Act_{LR}(q) > Act_{LR}(p)$. Moreover, if $Reach_{LR}(p, P)$ is defined, then $P \geq Act_{LR}(p)$. Note that if $P = Act_{LR}(p)$, then $Reach_{LR}(p, P) = Pos[p] - P$ and thus, $Reach_{RL}(p+1, P) \geq Pos[p+1] - P > Pos[p] - P \geq Reach_{LR}(p, P)$. This contradicts the first claim of the lemma.

For similar reasons, for each $q \in [p+1, n]$, $Act_{RL}(q) < P$.

We now prove that for each $q \in [1, p]$, $P < TH_{RL}(q)$. Suppose there exists such a q and consider $LR = \{a_r \mid r \leq q-1\}$ and $RL = \{a_r \mid r \geq q\}$. Since $P > Act_{LR}(q)$, $Reach_{LR}(q-1, P) > Pos[q] - P = Reach_{RL}(q, P)$ and consequently, from the first claim of the lemma, it implies that there exists $P' < P$ such that $Reach_{LR}(q-1, P') > Reach_{RL}(q, P')$. This would imply that P is not the optimal value needed to solve convergecast.

For similar reasons, for each $q \in [p+1, n]$, $P < TH_{LR}(q)$. □

B Algorithm `ComputeOptimal`

Function `ComputeOptimal(array` $Pos[1:n]$ `of real):real`

$\mathtt{TH_{LR}} =$ **empty_stack;** $\mathtt{TH_{RL}} =$ `ThresholdRL`(Pos)**;**
$(q, P_{RL}) =$ **pop**$(\mathtt{TH_{RL}})$**;** $P_{OPT} = P_{RL}$**;**
/* $q = 1$, $P_{RL} = TH_{RL}(1)$ */
$(q, P_{RL}) =$ **pop**$(\mathtt{TH_{RL}})$**;** $p = 1$**;** $P_{LR} = 0$**;**
for $r = 1$ **to** $n - 1$ **do**
  /* $P_{OPT} = P_{<r} \geq P_{LR}, P_{RL}$ */
  if $2^{r-p}Pos[p] + (2^{r-p+1} - 1)P_{OPT} - S_{LR}(p, r) > 2^{q-r-1}Pos[q] - (2^{q-r} - 1)P_{OPT} - S_{RL}(q, r + 1)$ **then**
    /* If $Reach_{LR}(r, P_{OPT}) > Reach_{RL}(r + 1, P_{OPT})$ then P_{OPT} is larger than the value needed to solve convergecast at position r. We apply now the same algorithm as in function OptimalAtIndex. */
    $P = \max\{P_{LR}, P_{RL}\}$**;**
    while $2^{r-p}Pos[p] + (2^{r-p+1} - 1)P - S_{LR}(p, r) \geq 2^{q-r-1}Pos[q] - (2^{q-r} - 1)P - S_{RL}(q, r + 1)$ **do**
      if $P_{LR} \geq P_{RL}$ **then** $(p, P_{LR}) =$ **pop**$(\mathtt{TH_{LR}})$**;**
      else $(q, P_{RL}) =$ **pop**$(\mathtt{TH_{RL}})$**;**
      $P = \max\{P_{LR}, P_{RL}\}$**;**
    $P_{OPT} = (2^{q-r-1}Pos[q] - S_{RL}(q, r + 1) - 2^{r-p}Pos[p] + S_{LR}(p, r))/(2^{r-p+1} + 2^{q-r} - 2)$**;**
    /* $P_{OPT} = P_{<r+1}$ is the solution of $Reach_{LR}(r, P_{OPT}) = Reach_{RL}(r + 1, P_{OPT})$ */
  if $q = r + 1$ **then return** P_{OPT}**;**
  /* In this case, $P_{OPT} \geq TH_{RL}(r + 1)$ and thus $P_{OPT} = P_{<r} = Act_{LR}(r + 1)$: for any $s > r$, $P_{<s} = P_{<r}$ */
  if $2^{r-p} * Pos[p] + (2^{r-p+1} - 1) * P_{OPT} - S_{LR}(p, r) \geq Pos[r + 1]$ **then**
    /* If $Reach_{LR}(r, P_{OPT}) \geq Pos[r + 1]$ then $TH_{LR}(r + 1) \leq P_{OPT}$ and we update $\mathtt{TH_{LR}}$, using the same algorithm as in function ThresholdLR. */
    while $2^{r-p} * Pos[p] + (2^{r-p+1} - 1) * P_{LR} - S_{LR}(p, r)$ **do** $(p, P_{LR}) =$ **pop**$(\mathtt{TH_{LR}})$**;**
    push $(\mathtt{TH_{LR}}, (p, P_{LR}))$**;**
    $P_{LR} = (Pos[r + 1] + S_{LR}(p, r) - 2^{r-p} * Pos[p])/(2^{r-p+1} - 1)$**;**
    $p = r + 1$**;**

C Proofs of Section 3

Proof of Theorem 2 : We first prove the following claim.

Claim 5 *Consider a configuration (G, A). We have*

$$D(G,A) = \max_{\emptyset \subsetneq X \subsetneq A} \{ \min_{x \in X, y \in A \setminus X} \{d_G(x,y)\}\} \leq 2P_{OPT}$$

Suppose, by contradiction, that there is a partition of A into X and $A \setminus X$ such that for each $x \in X$ and $y \in A \setminus X$ the distance between x and y is greater than $2P_{OPT}$. It means that no agents in X can meet an agent in $A \setminus X$ with power P_{OPT}. This contradicts the fact that there is a convergecast strategy in G using battery power P_{OPT}. Hence, for every partition of A into X and $A \setminus X$, there exists agents $x \in X$ and $y \in A \setminus X$ that are at distance at most $2P_{OPT}$. This ends the proof of the claim.

First, we show that if each agent executes Algorithm 1 then, eventually, one agent will possess all the information. Consider an agent a executing the algorithm. Let $T_a(t)$ be the subtree rooted at the last visited node and containing all nodes accessible from its current position by shortest paths containing a non-null part of the last edge traversed. Hence, when a enters a new node u, u is added to T_a. We show by induction on the number of nodes of T_a that a has the initial information of every agent that started in T_a. For $|T_a| = 1$, it is true since a is the only agent that started in T_a. The size of T_a grows only when a enters or exits some node v. When a enters a new node v, we show that any agent that started at v did not move yet. Assume by contradiction that there is an agent b that started at v and has moved before the arrival of a. It means that agents have arrived from all but one edge incident to v. In that case, agent b follows the edge from which no agent has arrived. Hence, the only possible edge that agent b can follow is the edge taken by agent a to arrive at v. This leads to a contradiction since agents a and b must have met inside the edge and agent a would have stopped before reaching v. When an agent a moves from a node v of degree δ, there were $\delta - 1$ agents: $b_1, b_2, \ldots, b_{\delta-1}$ that have arrived at v. By induction hypothesis, each agent b_i, for $1 \leq i \leq \delta - 1$, has collected all the information from agents starting inside the subtree T_{b_i}. Since agent a moves to the only direction where no agent has arrived, it has the information of every agent that started in $T_a = T_{b_1} \cup T_{b_2} \cup \cdots \cup T_{b_{\delta-1}}$.

Observe that for each $1 \leq i \leq k$ the tree T_{a_i} grows until either a_i meet agents that have arrived from all incoming ports of its position, or another agent a_j with more power is chosen to move to an unexplored direction. In the later case, $T_{a_i} \subseteq T_{a_j}$ and the tree T_{a_j} will grow under the same conditions. Thus, $\cup_{i=1}^{k} T_{a_i}$ will eventually be equal to T. When this happens, either two agents u_1, u_2 meet at an edge or δ agents $u_1, u_2, \ldots, u_\delta$ meet on a node of degree δ. These agents have the entire information since $T_{u_1} \cup T_{u_2} \cup \cdots \cup T_{u_\delta} = T$ ($\delta = 2$ if the meeting occurs on an edge).

It remains to show that the agents do not use more battery power than $D(T, A)$. Suppose by contradiction, that an agent a uses battery power $P > D(T, A)$. Let p be the point where agent a has finished the execution of the algorithm (when the value of $collecting$ becomes false) and let v be the last node visited by a before reaching p. Consider T_a when a exited v. Agent a is the agent starting in T_a for which the distance between its initial position and the node v was the closest since it was the agent that has used the least power when it arrived at v. Thus, the distance between the initial position of an agent in T_a and an agent in $G \setminus T_a$ is greater or equal than P. We have $D(T, A) < P \leq D(T, A)$, a contradiction. By Claim 5, we have that $D(T, A) \leq 2P_{OPT}$ and the algorithm is 2-competitive. □

Before proving Theorem 3, we first prove two lemmas that we need in our proof of the theorem.

Lemma 6 *Consider any $\varepsilon > 0$, an amount of power P, and a set $\{a_0, a_1, \ldots, a_k, a_{k+1}\}$ of $k+2$ agents located at positions $Pos[0:k+1]$. If $Pos[k+1] - Reach_{LR}(0,P) \leq P - \varepsilon$, and if $k \geq \log(P/\varepsilon)$, there exists $i \leq k$ such that $Reach_{LR}(i,P) \geq Pos[i+1]$.*

Proof : Suppose, by contradiction, that the lemma does not hold. It means that for each $0 \leq i \leq k$, $Reach_{LR}(i,P) < Pos[i+1]$. Therefore, we have

$$\begin{aligned} Reach_{LR}(k,P) &= 2^k Reach_{LR}(0,P) + (2^k-1)P - \Sigma_{i=1}^{k} 2^{k-i} Pos[i] \\ &= Reach_{LR}(0,P) + (2^k-1)P - \Sigma_{i=1}^{k} 2^{k-i}(Pos[i] - Reach_{LR}(0,P)) \\ &\geq Reach_{LR}(0,P) + (2^k-1)P - \Sigma_{i=1}^{k} 2^{k-i}(P-\varepsilon) \\ &\geq Reach_{LR}(0,P) + (2^k-1)P - (2^k-1)(P-\varepsilon) \\ &\geq Reach_{LR}(0,P) + (2^k-1)\varepsilon \end{aligned}$$

Consequently, if $k \geq \log(P/\varepsilon)$, we have $Reach_{LR}(k,P) \geq Reach_{LR}(0,P) + P - \varepsilon \geq Pos[k+1]$, a contradiction. □

Lemma 7 *Consider an amount of power P, a distance $d > 0$, and a set $\{a_0, a_1, \ldots, a_k\}$ of k agents located at positions $Pos[1:k]$. Let R_0 be the closest point from $Pos[1]$ that a_0 reached. Assume that $Pos[1] - R_0 = d$.*

Suppose that all the agents execute the same deterministic algorithm and do not know their initial position, and assume that some agent $a \in \{a_1, a_2, \ldots, a_k\}$ meets agent a_0 before any agent $a' \in \{a_1, a_2, \ldots, a_k\}$ meets any other agent. Then, $a = a_1$ and when a_1 meets a_0, for each $1 \leq i \leq k$, agent a_i is located on $Pos[i] - d$.

Moreover, if R_{max} is the furthest point on the right reached by some agent knowing the initial information of agent a_0, then $R_{max} - Pos[k] \leq P - 2d$.

Proof : Since all agents are executing the same deterministic algorithm, let us consider the execution of the algorithm until some agent reaches agent a_0. During this period, all the agents perform exactly the same moves and thus no agent meets another agent before agent a_1 meets a_0 at point R_0 or on the left of R_0. When agent a_1 meets a_0, it has moved at least a distance of d. Until this meeting between a_0 and a_1, every other agent has also moved a distance of at least d, and is located at distance d to the left of its starting position. Consequently, no agent can go further than $P - 2d$ to the right of $Pos[k]$. □

Proof of Theorem 3 : Let $\varepsilon = \delta P/4$ and $\sigma = \varepsilon/2 = \delta P/8$. Let $l = \lfloor \log(8/\delta) \rfloor$ and $k = l + 2$.

Consider a set of agents positioned on a line as follows (See Figure 1). There is an agent a_0 (resp. a_{2l+1}) on the left (resp. right) end of the line on position $s'_0 = 0$ (resp. s_l). For each $1 \leq i \leq 2l$, there is a set A_i of k agents that start on distinct positions within a segment $[s_i, s'_i]$ of length σ such that for each $1 \leq i \leq 2l+1$, the distance between s_i and s'_{i-1} is $2(P-\varepsilon)$. In other words, for each i, $s_i = (2P-3\sigma)i - \sigma$ and $s'_i = (2P-3\sigma)i$.

First, let us consider the execution of the optimal centralized algorithm for this configuration. We claim that if the amount of power given to each agent is P, then convergecast is achievable. We show by induction on i that for every i, $Reach_{LR}(ik,P) \geq s'_i + P - \epsilon = s_{i+1} - P + \epsilon$. For $i = 0$, $Reach_{LR}(0,P) = Pos[0] + P > P - \epsilon = s_1 - P + \epsilon$. Suppose that $Reach_{LR}((i-1)k,P) \geq s_i - P + \epsilon$. Consider the agents in A_i, i.e., the agents a_{ik-j}, $j \in [0, k-1]$. Since $s'_i - Reach_{LR}((i-1)k,P) \leq P - \epsilon + \sigma = P - \sigma < P$, and since $l+1 \geq \log(P/\sigma)$, we know by Lemma 6 that $Reach_{LR}(k(i-1)+l+1,P) \geq Pos[k(i-1)+l+2]$. Since $k > l+1$, it follows that $Reach_{LR}(ik,P) = Pos[ik] + P \geq s_i + P = s'_i + P - \sigma \geq s'_i + P - \epsilon$. Consequently,

the induction hypothesis holds. Since $Reach_{LR}(2lk, P) \geq s_{2l+1} - P + \epsilon \geq Reach_{RL}(2lk+1, P)$, P is sufficient to solve convergecast.

Suppose now that there exists a distributed strategy $\mathcal{S}$ that solves convergecast on the configuration when the amount of power given to each agent is $(2-\delta)P$. Consider an execution of $\mathcal{S}$. A *step* in the execution of $\mathcal{S}$ is a moment where two agents meet. Let tl_i (resp. tr_i) be the first step where an agent from A_i meets an agent from $A_{i'}$ with $i' < i$ (resp. $i' > i$). Let R_i (resp. L_i) be the furthest point on the right (resp. on the left) reached by any agent from A_i once some agent in A_i has met an agent from $A_{i'}$ with $i' < i$ (resp. $i' > i$). For any $1 \leq i < j \leq 2l+1$, let $A_{i,j} = A_i \cup A_{i+1} \ldots \cup A_j$.

We show by induction on t that for each $i \in [1, l]$ such that $tl_i \leq t$ and for each $j \in [l+1, 2l]$ such that $tr_j \leq t$, the following holds:

- (i) $tl_i < tl_{i'}$ for each $i' \in [i+1, l]$ and $tr_j < tr_{j'}$ for each $j' \in [l+1, j-1]$,
- (ii) for each $i' \in [i+1, l]$, if $tl_{i'} > t$ then $tr_{i'} > t$, and for each $j' \in [l+1, j-1]$, if $tr_{j'} > t$ then $tl_{j'} > t$
- (iii) $R_i \leq s_{i+1} - (2^{i+2} - 2)\varepsilon$ and $L_j \geq s'_{j-1} + (2^{2l-1-j} - 2)\varepsilon$,
- (iv) no agent in $A_{i'}$, $i' \geq l$ meets any agent from $A_{1,l-1}$ and no agent in $A_{j'}$, $j' \leq l+1$ meets any agent from $A_{l+2,2l+1}$.

Clearly, $R_0 \leq s'_0 + 2P - \delta P = 2P - 4\varepsilon = s_1 - 2\varepsilon$ and $L_{2l+1} \geq s_{2l+1} - 2P + \delta P = s_{2l} + 2\varepsilon$. Since all agents in $A_{1,2l}$ execute the same algorithm, they all perform the same moves until either the leftmost agent of A_1 meets a_0 (at step tl_1), or the rightmost agent of A_{2l} meets a_{2l+1} (at step tr_{2l}). In the first case, it shows that $tl_1 < tl_i, tr_i$ for any $i \geq 2$, and by Lemma 7, $R_1 \leq s'_1 + (2-\delta)P - 2(s_1 - R_0) \leq s_2 - 2P + 2\varepsilon + 2P - 4\varepsilon - 2(2\varepsilon) = s_2 - 6\varepsilon$. By symmetry, in the second case, $tr_{2l} < tr_i, tl_i$ for any $i \leq 2l-1$ and $L_{2l} \geq s'_{2l-1} + 6\varepsilon$. In both cases, the induction hypothesis holds.

Suppose now that the induction hypothesis holds for all $t' < t$ and let $i = \max\{i' \mid tl_{i'} < t\} + 1$ and $j = \min\{j'+1 \mid tr_{j'} < t\} - 1$. Note that by (iv), we have $i \leq l-1$ and $j \geq l+2$. By (i) and (ii), before step t, no agent in $A_{i'}$, $i \leq i' \leq j$ has met any other agent from a set $A_{i''}$, $i' \neq i''$. Thus, since all agents in A_{ij} execute the same algorithm, they have performed exactly the same moves and they have not met any other agent before step t. Suppose that an agent from A_{ij} meets another agent at step t. Then, either the leftmost agent a_i from A_i meets an agent $a_{i'}$ from $A_{i'}$ with $i' < i$, or the rightmost agent from A_j meets an agent from $A_{j'}$ with $j' > j$.

By symmetry, it is enough to consider only one case. In the following, we assume that $a_i \in A_i$ meets an agent $a_{i'} \in A_{i'}$ with $i' < i$ at step t. In this case, $t = tl_i$ and thus $tl_i < tl_{i'}, tr_{i'}$ for each $i < i' \leq j$; consequently, (i) and (ii) hold. Moreover, by induction hypothesis, the meeting between a_i and $a_{i'}$ occurs at a point $p \leq R_{i'} \leq R_{i-1} \leq s_i - (2^{i+1} - 2)\varepsilon$. Suppose first that $i < l-1$. By Lemma 7, we have $R_i \leq s'_i + 2P - \delta P - 2(2^{i+1} - 2)\varepsilon = s_{i+1} - 2P + 2\varepsilon + 2P - 4\varepsilon - 2^{i+2}\varepsilon + 4\varepsilon = s_{i+1} - (2^{i+2} - 2)\varepsilon$, and thus (iii) holds. Suppose now that $i = l \geq \log(8/\delta) - 1$. Then $R'_i \leq s_l - (8/\delta - 2)\delta P/4 = s_l - 2P + \delta P/2 < s_l - 2P + \delta P$. But this is impossible since the initial position of the leftmost agent a of A_l is $Pos[lk+1] \geq s_l$ and the power available to a is $2P - \delta P$, hence (iv) holds.

Therefore, no agent from $A_{1,l}$ ever meets any agent from $A_{l+1,2l+1}$ and consequently, $\mathcal{S}$ is not a distributed convergecast strategy. □

D Proofs of Section 4

We consider the centralized convergecast decision problem formalized as follows.

Centralized convergecast decision problem
Instance: a weighted graph G, a set of k agents located at nodes of G and a real P_{max}.
Question: Is there a strategy solving the convergecast for the agents in graph G in which each agent uses at most P_{max} battery power ?

To prove Theorem 4, we first show that the centralized convergecast decision problem is strongly NP-hard and then that the problem is in NP.

Lemma 8 *The centralized convergecast decision problem is strongly NP-hard for trees.*

Proof : We construct a polynomial-time many-one reduction from the following strongly NP-Complete problem [27].

3-Partition problem
Instance: a multiset S of $3m$ positive integers x_i such that for $1 \leq i \leq 3m, R/4 < x_i < R/2$ with $R = \frac{\sum_{i=1}^{3m} x_i}{m}$.
Question: Can S be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ of size three such that for $1 \leq j \leq m$, $\sum_{x \in S_j} x = R$?

We construct an instance (G, U) of the centralized convergecast problem from an instance of 3-Partition as follows. The graph G is a star with $4m+2$ leaves and U is the set of leaves of G. Hence, there are $4m+2$ agents each located at a leaf of the star. We consider a partition of the set of agents into three subsets: A, B and C. The subset $A = \{a_i \mid 1 \leq i \leq m+1\}$ contains $m+1$ agents. The leaves containing these agents are incident to an edge of weight 1. The subset $B = \{b_i \mid 1 \leq i \leq 3m\}$ contains $3m$ agents. For $1 \leq i \leq 3m$, the weight of the edge incident to the leaf containing agent b_i is $2R + 1 + x_i$. The subset $C = \{c\}$ contains one agent. The leaf containing agent c is incident to an edge of weight $4R + 1$. Figure 2 depicts the star obtained. The battery power P_{max} allocated to each agent is equal to $2R + 1$. The construction can be done in polynomial time. We show that the constructed instance of the centralized convergecast problem has a solution if and only if the original instance of 3-partition has a solution.

First, assume there exists a solution $S_1, S_2, \ldots, S_m$ for the instance of the 3-partition problem. We show that the agents can solve the corresponding instance of the centralized convergecast problem using the following strategy. Agent c moves at distance $2R$ of the center and for each $1 \leq i \leq 3m$, agent b_i moves at distance x_i from the center. At this point, all these agents have use all their battery power. Each agent in A moves to the center of the star. For $1 \leq i \leq m$ and each of the three agents b_j such that $x_j \in S_i$, agent a_i moves to meet b_j and go back to the center of the star. The cost of this movement is $2 \sum_{x_j \in S_i} = 2R$, which is exactly the remaining battery power of agent a_i. Observe that since agents in A have met all agents in B, agents in A, located at the center of the star, have the information of all agents except agent c. Agent A_{m+1} then moves to meet agent c. Agents a_{m+1} and c have the information of all the agents. Hence, this is a solution to the instance of the centralized convergecast problem.

Now, assume there is no solution to the instance of 3-partition problem. Suppose by contradiction that there is a solution to the corresponding instance of the centralized convergecast problem. For the sake of simplicity, we show first that we can only consider strategy with special properties. A convergecast strategy in a star is called *simple* if :

- The strategy starts with a gathering phase in which each agent uses all their available power to move to the center of the star and then waits until time P_{max}. The agent that have used all their power during this phase are called *depleted*.
- The agents does not move past depleted agents, i.e., never enters the segment between a leaf and a depleted agent on the incident edge.

The following claim shows that we can only consider simple convergecast strategy.

Claim 6 *If there is convergecast strategy in a star using power P then there is a simple convergecast strategy using less power than P.*

Let $\mathcal{S}$ be the existing convergecast strategy. We construct a simple strategy $\mathcal{S}'$ as follows. In $\mathcal{S}'$, each agent moves towards the center of the star until it has used all its battery power or has reached the center of the star. This gathering phase lasts from time 0 to time P. If an agent has reached the center at time t in strategy $\mathcal{S}$, it executes at time $t' + P$ in strategy $\mathcal{S}'$ each movement performed at time $t' \geq t$ in strategy $\mathcal{S}$. However, if a movement of an agent would result in the agent moving past a depleted agent from time r to r' in $\mathcal{S}$, then the agent waits at the position of the depleted agent instead of moving from time r to r'. By construction, $\mathcal{S}'$ is a simple convergecast strategy. Observe that in strategy $\mathcal{S}'$, the non-depleted agents share all their information at the center of the star at time P_{max}. Moreover, if two depleted agents meet at time t in $\mathcal{S}$, they also meet in $\mathcal{S}'$ at time t'. Since two depleted agents cannot meet, it remains to show that when a non-depleted agent a meet a depleted agent b at time t in strategy $\mathcal{S}$, they meet at time $t + P$ in $\mathcal{S}'$. The position of agent a is closer to the center in $\mathcal{S}'$ than in $\mathcal{S}$. Hence, any agent b that meets agent at time t is at the new position of a in $\mathcal{S}'$ at time $t + P$.

By Claim 6, we can assume that the convergecast strategy is simple. Consider the star G after the gathering phase of the simple strategy. Each agent in A is at the center of the star. For $1 \leq i \leq m + 1$, the agent a_i has the remaining power of $2R$. For $1 \leq i \leq 3m$, the agent b_i is at distance x_j from the center of the star and agent c is at distance $2R$ from the center. Since the agents in A are the only agents with remaining battery power, they must move to collect the information of agents in $B \cup C$. We call this phase the collecting phase. Observe that since agent c is at distance $2R$ from the center, it is impossible for agents in A to move this information. Indeed, when an agent reaches c, it has used all its battery power. Hence, the entire information must be collected at the position of c. In order to collect the information, agents in A must go to the position of each agent in B and transport the information of these agents to the center. The total cost to move these information is at least twice the sum of the distances between each agent in B and the center. This is equal to $2\sum_{i=1} 3mx_i = 2Rm$. Then, these information must then be moved to the position of c. This costs at least $2R$. Hence, the total cost in order to collect the information after the gathering phase is at least $2R(m + 1)$. The amount of power available to the agents for the collecting phase is equal to the minimal amount of power to collect the information, since there are $m+1$ agents having each $2R$ power. This means that during the collecting phase, for $1 \leq j \leq 3m$, agents cannot use more than $2x_i$ power to collect the information of b_i.

Suppose by contradiction that during the collecting phase, more than one agent in A enters an edge f to collect the information of agent b_i at distance x_i from the center for some i s.t. $1 \leq i \leq 3m$. Let w be the agent that has reached the position of b_i. If w comes back to the center, it has used at least $2x_i$ power. Since at least one other agent has used some power to enter edge f, these agents has used more than $2x_i$ battery power to collect information of agent b_i. If w does not come back to the center then some other agent has to move the information to the center. If the agent w stops at distance r from the center then at least one other agents have to go to this position (at distance r from the center) and come back. Thus, the cost is at least $(2x_i - r) + 2r > 2x_i$. In both cases, the agents have used more $2x_i$ power, which leads to a contradiction. Hence, for each $1 \leq i \leq 3m$, there is only one agent that collects the information of agent b_i and enters the corresponding edge.

We can assume, without loss of generality, that agent a_{m+1} is the agent that move the information to c. Observe that a_{m+1} cannot collect information from other nodes since moving to c uses exactly all its remaining power. Hence, only agents in $A' = A \setminus \{a_{m+1}\}$ can collect these information of agents in C. Let $S_1, S_2, \ldots S_m$ be the partition of S defined by $S_i = \{x_j \mid \text{the information of } b_j \text{ is collected by } a_i\}$, for each $1 \leq i \leq m$. We have $2\sum_{x \in S_i} x \leq 2R$ since the agents only have $2R$ battery power. The power needed to

collect information of agents in B is $2mR$ which is exactly equal to the combined power available to agents in A'. This means that each agent in A' must use all their power to collect information and $2\sum_{x\in S_i} x = 2R$. Hence, $S_1, S_2, \ldots S_m$ is a solution to the instance of 3-partition, a contradiction. Thus, there is no solution to the centralized convergecast problem. □

Lemma 9 *The centralized convergecast decision problem is in NP.*

Proof : We consider the verifier-based definition of NP. Consider the strategy $\mathcal{S}$ of the agents for an instance of the centralized convergecast problem. We construct the certificate for the instance as follows. We say that a meeting of two or more agents is *useful* if at least one of the agents received a new piece of information during this meeting. Each agent participates in at most $k-1$ useful meetings. Hence, there are at most $k(k-1)$ useful meetings. The certificate contains the list of all useful meetings in chronological order. For the i-th meeting, the certificate encodes the identities of the meeting agents and the location of the meeting: a node x_i or an edge (u_i, v_i) of the graph G. If the meeting has occurred on an edge, the certificate encodes a variable d_i. The variable d_i represents the distance between u_i and the meeting point p_i. If a previous meeting of number j has occurred on the same edge, the certificate encodes if $d_i < d_j$, or $d_i = d_j$ or $d_i > d_j$. For each of the meeting agents, the certificate also encodes the node from which it has entered the edge (u_i or v_i) just before the meeting and the node from which it exits the edge just after the meeting. We consider the strategy $\mathcal{S}'$ defined as follows. For each useful meeting in chronological order, the meeting agents move to the meeting location following a shortest path from their previous position. If the meeting occurs on an edge, the meeting agents enter and exit the edge using the node encoded in the certificate. $\mathcal{S}'$ is a convergecast strategy since each time an agent has collected a new information in $\mathcal{S}$, it collects the same information during the corresponding meeting in $\mathcal{S}'$. Moreover, the agents use at most as much power in $\mathcal{S}'$ than in $\mathcal{S}$ since they move to the same meeting points using shortest paths. The verifier simulates the strategy $\mathcal{S}'$ defined by the certificate. The verifier first checks that all the agents possess the entire information at the end of the algorithm. This can be done in polynomial time. Then, the verifier computes the distance traveled by each agent. These distances are linear sums of variables d_i with $1 \leq i \leq k(k-1)$ and a constant. Finding an assignation of the variables, such that the distance traveled by each agent is less or equal than P_{max}, can be done in polynomial time using linear programming. Thus, the certificate can be verified in polynomial time. □

Proof of Theorem 5 : The parameters of algorithm `KnownGraph` are the graph G and the nodes corresponding to the initial positions of agents (stored in $Pos[1:k]$).

Algorithm 2: KnownGraph(a weighted graph G, an array Pos[1:k] of nodes)

```
strategy=empty_stack;
V:={Pos[1]};
P:=0;
repeat
    choose a couple (u,v) ∈ V × (Pos \ V) such that d(u,v) is minimal;
    V := V ∪ {v};
    Path := shortest path between u and v;
    push(strategy, (v, Path, u));
    P = max{P, d(u,v)};
until V = Pos;
repeat
    (v, Path, u) = pop(strategy);
    agent starting in u moves to v following path Path;
until strategy = empty_stack;
```

Let (u_i, v_i) be the nodes chosen at the i-th iteration of the first loop and let V_i be the value of V at the end of the i-th iteration. We set $u_0 = Pos[1]$ and $V_0 = \{u_0\}$. We show, by induction, that at the start of the i-th iteration of the second loop, every information is possessed by an agent that started in V_{k-i}. It is clearly true for $i = 1$. Assume by induction that it is true for i. Agent at u_{k-i} move to an agent at $v_{k-i} = u_{k-j}$ for some $j > i$, during the i-th iteration of the second loop. After this movement, agent in $V_{k-(i+1)}$ have all the information since $v_i \in V_{k-(i+1)}$. hence the property is true for $i + 1$ and so for all i. At the end of the algorithm, the agent at $Pos[1]$ have all the information since $V_0 = \{Pos[1]\}$.

Let A be the set of agents. Consider the partition of Pos into the sets V_{i-1} and $Pos \setminus V_{i-1}$. We have $d(u_i, v_i) \leq D(G, Pos)$ since (u_i, v_i) is the couple $(u, v) \in V_{i-1} \times (Pos \setminus V_{i-1})$ such that $d(u, v)$ is minimal. Hence, an agent will never move the distance longer than $2P_{OPT}$ by Claim 5. □