Journal of Computer and System Sciences 82 (2016) 802–816

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Bounds on the cover time of parallel rotor walks ☆

Dariusz Dereniowski [a], Adrian Kosowski [b], Dominik Pająk [c,*], Przemysław Uznański [d]

[a] Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland
[b] GANG Project, Inria Paris and IRIF, Paris, France
[c] Department of Computer Science, Faculty of Fundamental Problems of Technology, Wrocław University of Technology, Wrocław, Poland
[d] Department of Computer Science, ETH Zürich, Switzerland

ARTICLE INFO

Article history:
Received 18 December 2014
Received in revised form 11 January 2016
Accepted 29 January 2016
Available online 9 March 2016

Keywords:
Distributed graph exploration
Rotor-router
Cover time
Collaborative robots
Parallel random walks
Derandomization

ABSTRACT

The rotor-router mechanism was introduced as a deterministic alternative to the random walk in undirected graphs. In this model, a set of k identical walkers is deployed in parallel, starting from a chosen subset of nodes, and moving around the graph in synchronous steps. During the process, each node successively propagates walkers visiting it along its outgoing arcs in round-robin fashion, according to a fixed ordering. We consider the cover time of such a system, i.e., the number of steps after which each node has been visited by at least one walk, regardless of the initialization of the walks. We show that for any graph with m edges and diameter D, this cover time is at most $\Theta(mD/\log k)$ and at least $\Theta(mD/k)$, which corresponds to a speedup of between $\Theta(\log k)$ and $\Theta(k)$ with respect to the cover time of a single walk.

1. Introduction

In graph exploration problems, a walker or group of walkers (agents) is placed on a node of a graph and moves between adjacent nodes, with the goal of visiting all the nodes of the graph. The study of graph exploration is closely linked to central problems of theoretical computer science, such as the question of deciding if two nodes of the graph belong to the same connected component (*st-connectivity*). For example, fast approaches to connectivity testing in little memory rely on the deployment of multiple random walks [6,13]. In these algorithms, the initial locations of the walkers are chosen according to a specific probability distribution.

More recently, multiple walks have been studied in a worst-case scenario where the k agents are placed on some set of starting nodes and deployed in parallel, in synchronous steps. The considered parameter is the *cover time* of the process, i.e., the number of steps until each node of the graph has been visited by at least one walker. Alon et al. [2], Efremenko and Reingold [11], and Elsässer and Sauerwald [12] have studied the notion of the *speedup* of the random walk for an undirected graph G, defined as the ratio between the cover time of a k-agent walk in G for worst-case initial positions of agents and that of a single-agent walk in G starting from a worst-case initial position, as a function of k. A characterization of the

☆ Research partially supported by ANR project DISPLEXITY and by NCN under contract DEC-2011/02/A/ST6/00201. Dariusz Dereniowski was partially supported by a scholarship for outstanding young researchers funded by the Polish Ministry of Science and Higher Education. Some of the results of this paper appeared in the extended abstract [9], published in the Proceedings of the 31st Symposium on Theoretical Aspects of Computer Science (STACS 2014).

* Corresponding author.
E-mail address: pajakd@gmail.com (D. Pająk).

http://dx.doi.org/10.1016/j.jcss.2016.01.004
0022-0000/

speedup has been achieved for many graph classes with special properties, such as small mixing time compared to cover time. However, a central question poised in [2] still remains open: what are the minimum and maximum values of speedup of the random walk in arbitrary graphs? The smallest known value of speedup is $\Theta(\log k)$, attained e.g. for the cycle, while the largest known value is $\Theta(k)$, attained for many graph classes, such as expanders, cliques, and stars.

In this work, we consider a deterministic model of walks on graphs, known as the *rotor-router*. The rotor-router model, introduced by Priezzhev et al. [22], provides a mechanism for the environment to control the movement of the agent deterministically, mimicking the properties of exploration as the random walk. In the rotor-router, the agent has no operational memory and the whole routing mechanism is provided within the environment. The edges outgoing from each node v are arranged in a fixed cyclic order known as a *port ordering*, which does not change during the exploration. Each node v maintains a *pointer* which indicates the edge to be traversed by the agent during its next visit to v. If the agent has not visited node v yet, then the pointer points to an arbitrary edge adjacent to v. The next time when the agent enters node v, it is directed along the edge indicated by the pointer, which is then advanced to the next edge in the cyclic order of the edges adjacent to v. In this paper we also consider a class of processes called *fair strategies* which are a generalization of the rotor-router model. In a fair strategy an agent entering a node can choose an arbitrary outgoing arc among arcs with the minimum number of traversals.

For a single agent, the (deterministic) cover time of the rotor-router and the (expected) cover time of the random walk prove to be surprisingly convergent for many graph classes. In general, it is known that for any n-node graph of m edges and diameter D, the cover time of the rotor-router in a worst-case initialization is precisely $\Theta(mD)$ [26,3]. By comparison, the random walk satisfies an upper bound of $O(mD \log n)$ on the cover time, though this bound is far from tight for many graph classes. Different locally fair exploration strategies were considered in [7]: Oldest-First and Least-Used-First. In the Oldest-First strategy, an agent visiting a node chooses an edge that has not been traversed for the longest time. In the Least-Used-First strategy it chooses an edge with the smallest number of traversals. In both strategies ties can be broken in an arbitrary way. Cooper et al. [7] showed that in undirected graphs any Oldest-First strategy achieves cover time of $O(mD)$ whereas exploration using Least-Used-First strategy can lead to exponential cover time. Note that in directed symmetric graphs, the Oldest-First strategy is equivalent to the rotor-router and the Least-Used-First is more general and is equivalent to the class of fair strategies. Yanovski et al. [26] observed, based on the analysis of Koenig and Simmons [17], that any fair strategy has a cover time of $O(mD)$.

The behavior of the rotor-router model with multiple agents appears to be much more complicated. Since the parallel walkers interact with the pointers of a single rotor-router system, they cannot be considered independent (in contrast to the case of parallel random walks). In the first work on the topic, Yanovski et al. [26] showed that adding a new agent to a rotor-router system with k agents cannot increase the cover time, and showed experimental evidence suggesting that a speedup does indeed occur. Klasing et al. [16] have provided the first evidence of speedup, showing that for the special case when G is a cycle, a k-agent system explores an n-node cycle $\Theta(\log k)$ times more quickly than a single agent system.

In this work we completely resolve the question of the possible range of speedups of the parallel rotor-router model in a graph, showing that its value is between $\Theta(\log k)$ and $\Theta(k)$, for any graph. Both of these bounds are tight. Thus, the proven range of speedup for the rotor-router corresponds precisely to the conjectured range of speedup for the random walk. We also show that the speedup of at least $\Theta(\log k)$ is achieved for any fair strategy.

1.1. Related work

The rotor-router model Studies of the rotor-router started with works of Wagner et al. [25] who showed that in this model, starting from an arbitrary configuration (arbitrary cyclic orders of edges, arbitrary initial values of the port pointers and an arbitrary starting node) the agent covers all m edges of an n-node graph within $O(nm)$ steps. Bhatt et al. [5] showed later that within $O(nm)$ steps the agent not only covers all edges but enters (establishes) an Eulerian cycle. More precisely, after the initial stabilization period of $O(nm)$ steps, the agent keeps repeating the same Eulerian cycle of the directed symmetric version $\vec{G}$ of graph G (see Section 3 for a definition). Subsequently, Yanovski et al. [26] and Bampas et al. [3] showed that the Eulerian cycle is in the worst case entered within $\Theta(mD)$ steps in a graph of diameter D. Considerations of specific graph classes were performed in [14]. Robustness properties of the rotor-router were further studied in [4], where it has been considered the time required for the rotor-router to stabilize to a (new) Eulerian cycle after an edge is added or removed from the graph. Regarding the terminology, we note that the rotor-router model has also been referred to as the *Propp machine* [3] or *Edge Ant Walk algorithm* [25,26], and has also been described in [5] in terms of traversing a maze and marking edges with pebbles. Studies of the multi-agent rotor-router were performed by Yanovski et al. [26] and Klasing et al. [16], and its speedup was considered for both worst-case and best-case scenarios.

A variant of the multi-agent rotor-router mechanism has been extensively studied in a different setting, in the context of balancing the workload in a network. In such a scenario, the walks in the graph are performed by entities referred to as *tokens*. Cooper and Spencer [8] studied d-dimensional grid graphs and showed a constant bound on the discrepancy, defined as the difference between the number of tokens at a given node v in the rotor-router model and the expected number of tokens at v in the random-walk model. Subsequently, Doerr and Friedrich [10] analyzed in more detail the distribution of tokens in the rotor-router mechanism on the 2-dimensional grid. Akbari and Berenbrink [1] showed an upper bound of $O(\log^{3/2} n)$ on the discrepancy for hypercubes and a bound of $O(1)$ for a constant-dimensional torus.

Table 1
Values of speedup for k-agent exploration with the rotor-router and parallel random walks. All results hold at least in the range $k \leq n$.

Graph class	Speedup of Rotor-Router		Speedup of Random Walk			
	for cover time		*for cover time*		*for max hitting time*	
General case:	$\Omega(\log k)$, $O(k)$	(**Theorems 3.9, 4.1**)	$O(k^2)$, $O(k \log n)$	[11,12]	$O(k)$	[12]
Cycle:	$\Theta(\log k)$	[16,18]	$\Theta(\log k)$	[2]	$\Theta(\log k)$	[2]
Star:	$\Theta(k)$	(**Proposition 4.2**)	$\Theta(k)$	[2]	$\Theta(k)$	[2]

The rotor-router model can be generalized to serve as a derandomization of a Markov Chain with arbitrary probabilities by modifying the ordering of edges [15]. Holroyd and Propp [15] showed that a Markov Chain and its rotor-router analogue are close in terms of hitting frequencies, hitting times and occupation frequencies. An interesting application of the concept of the generalized rotor-router is presented in [24] where the authors show how to use multiple rotor walks to obtain an efficient deterministic sampler for some #P-complete problems, for example for 0-1 knapsack solutions, linear extensions, matchings, etc., for which rapidly mixing chains are known.

Another variant of the rotor walk is called *rotor-router aggregation* and was introduced by Jim Propp. In the rotor-router aggregation model, multiple tokens (particles) start at a specific vertex, known as the origin. Each token in turn performs a rotor walk, starting at the origin, until it reaches a node not occupied by any other token. The asymptotic shape of the set of occupied nodes for n particles in $\mathbb{Z}^d$ with a clockwise ordering of outgoing edges was studied by Levine and Peres [19–21].

Parallel random walks Alon et al. [2] introduced the notion of the speedup of k independent random walks as the ratio of the cover time of a single walk to the cover time of k random walks. They conjectured that the speedup is between $\log k$ and k for any graph. The speedup was shown to be k for many graph classes, such as complete graphs [2], d-dimensional grids [2,12], hypercubes [2,12], expanders [2,12], and different models of random graphs [2,12]. For the cycle, the speedup is equal to $\log k$ [2]. For general graphs, an upper bound of $\min\{k \log n, k^2\}$ on the speedup was obtained by Efremenko et al. [11]. Independently, Elsässer et al. [12] showed the $k \log n$ upper bound. For binary trees, Sauerwald [23] showed an upper bound on the cover time of $O\left(n \log^5 n/\sqrt{k}\right)$ and a lower bound that is within a polylogarithmic factor (in n) of this upper bound, providing evidence that the speedup on binary trees is $\sqrt{k}$.

Another measure studied by Efremenko et al. [11] concerns the speedup with respect to a different exploration parameter — the maximum hitting time, i.e., the maximum over all pairs of nodes of the graph of the expected time required by the walk to move from one node to the other. For this parameter, they showed a bound on speedup of $O(k)$, mentioning that it is tight in many graph classes.

1.2. Our results and overview of the paper

In this work we establish bounds on the minimum and maximum possible cover time for a worst-case initialization of a k-rotor-router system in a graph G with m edges and diameter D.

In Section 2 we provide a formal definition of the rotor-router model and of fair strategies, recalling their basic properties. In Section 3, we first prove that the cover time t_C for any fair strategy satisfies $t_C \in O(mD/\log k)$, when $k < 2^{16D}$. We then extend this result to the case of $k \in O(\text{poly}(n))$, i.e., we show that for any constant $c_1 > 0$ there exists a constant $c_2 > 0$, such that if $k < n^{c_1}$, then $t_C < c_2 mD/\log k$. The main part of our proofs relies on a global analysis of the number of visits to edges in successive time steps, depending on the number of times that these edges have been traversed in the past. We first prove a stronger version of local structural lemmas proposed by Yanovski et al. [26], and apply them within a global amortization argument over all time steps and all edges in the graph. The extension to the case of $k \in O(\text{poly}(n))$ relies on a variant of a similar amortized analysis, and also makes use of a technique known as *delayed deployments* introduced by Klasing et al. [16], which we briefly recall in Section 2. We remark that by [16,18], a cover time of $\Theta(mD/\log k)$ is achieved when G is a cycle with all agents starting from one node.

In Section 4, we show a complementary lower bound on the cover time of the k-agent rotor-router in worst case initialization, namely, $t_C \in \Omega(mD/k)$. As a starting point, the proof uses a decomposition of the edge set of a graph, introduced by Bampas et al. [3], into a "heavy part" containing a constant proportion of the edges and a "deep part", having diameter linear in D. The main part of the analysis is to show that an appropriate initialization of k agents in the heavy part takes a long time to reach the most distant nodes of the deep part. The argument also takes advantage of the delayed deployment technique. We close the section by remarking that a cover time of $\Theta(mD/k)$ is, in fact, achieved for some graphs, such as stars.

Table 1 contains a summary of our results on the speedup of the k-agent rotor-router, compared to corresponding results from the literature for parallel random walks. Note that for a deterministic process such as the rotor-router, the notions of cover time and maximum hitting are equivalent, and hence we only refer to cover times.

2. Model and preliminaries

Let $G = (V, E)$ be an undirected connected graph with n nodes, m edges and diameter D. We denote the neighborhood of a node $v \in V$ by $\Gamma(v)$. The directed graph $\vec{G} = (V, \vec{E})$ is the directed symmetric version of G, where the set of arcs $\vec{E} = \{(v,u), (u,v) : \{v,u\} \in E\}$. We will denote arc (v,u) by $v \to u$. For a node v of G, $\deg(v)$ is the number of edges incident to v in G. Given a subset $X \subseteq V$, $G[X]$ denotes the subgraph of G *induced by* X, $G[X] = (X, \{\{u,v\} \in E \mid u, v \in X\})$.

Definition of the rotor-router model We consider the rotor-router model (on graph G) with $k \geq 1$ indistinguishable agents, which run in steps, synchronized by a global clock. Each agent moves in discrete steps from node to node along the arcs of graph $\vec{G}$. A *configuration* at the current step is defined as a triple $((\rho^v)_{v\in V}, (\pi^v)_{v\in V}, \{r_1, \ldots, r_k\})$, where ρ^v is a cyclic order of the arcs (in graph $\vec{G}$) outgoing from node v, π^v is an arc outgoing from node v, which is referred to as *the (current) port pointer at node* v, and $\{r_1, \ldots, r_k\}$ is the (multi-)set of nodes currently containing an agent. For each node $v \in V$, the cyclic order ρ^v of the arcs outgoing from v is fixed at the beginning of exploration and does not change in any way from step to step.

For an arc $v \to u$, let $next(v \to u)$ denote the next arc after arc $(v \to u)$ in the cyclic order ρ^v. The exploration starts from some initial configuration and then keeps running in all future time steps, without ever terminating. During the current step, first each agent i is moved from node r_i traversing the arc π^{r_i}, and then the port pointer π^{r_i} at node r_i is advanced to the next arc outgoing from r_i (that is, π^{r_i} becomes $next(\pi^{r_i})$). This is performed sequentially for all k agents. Note that the order in which agents are released within the same step is irrelevant from the perspective of the system, since agents are indistinguishable. For example, if a node v contained two agents at the start of a step, then it will send one of the agents along the arc π^v, and the other along the arc $(v, next(\pi^v))$.

Definition of a fair strategy A fair strategy can be seen as a generalization of the rotor-router model, in which at each node the ordering of outgoing edges may change after each full rotation of the rotor. Formally, in a fair strategy, the agents move in discrete time steps, synchronized by a global clock. A *configuration* at a time step t is defined as the triple $((\rho^v)_{v\in V}, (\sigma^v)_{v\in V}, \{r_1, \ldots, r_k\})$, where each ρ^v is an infinite sequence of arcs (in graph $\vec{G}$) outgoing from node v, each $\sigma^v \in \{0, 1, 2, \ldots\}$ represents the index of the next arc in the sequence ρ^v to be used by an agent leaving v, and $\{r_1, \ldots, r_k\}$ is the (multi-)set of nodes currently containing an agent. Moreover, it is assumed that for each v, the sequence ρ^v satisfies the following fairness condition: for any $j = 0, 1, 2, \ldots$, the subsequence consisting of elements $\rho^v_{j\deg(v)}, \rho^v_{(j\deg(v)+1)}, \cdots, \rho^v_{(j\deg(v)+\deg(v)-1)}$ forms a permutation of the set $\{0, 1, \ldots, \deg(v) - 1\}$. The operation of a fair strategy is such that each of the k agents is sequentially released from the respective node r_i along the arc with port $\rho^{r_i}_{\sigma^{r_i}}$, and the value of σ^{r_i} is incremented by one. When multiple agents occupy the same vertex v, they are released in the same step to different nodes, and the value of σ^v is incremented multiple times in this step. Note that by specifying a configuration of a fair strategy on a graph at a given time step, the configurations of the system at all future time steps are uniquely determined. Hence, whenever this does not lead to misunderstanding, we will use the term *strategy* to refer to the configuration of the system at time step 0.

Notation for edge and node counters Throughout the paper, $\mathbb{N}_+$ denotes the set of positive integers, and $\mathbb{N} = \mathbb{N}_+ \cup \{0\}$. We introduce compact notation for discrete intervals of integers: $[a, b] \equiv \{a, a+1, \ldots, b\}$, and $[a, b) \equiv [a, b-1]$, for $a, b \in \mathbb{N}$.

For a fixed strategy, we will denote by $a^{(t)}(e)$ the number of agents traversing directed arc $e \in \vec{E}$ in step $t+1$. We recall that multiple agents traversing one arc $e \in \vec{E}$ in the same time step are considered to move simultaneously. By $d^{(t)}(e)$ we denote the number of traversals of directed arc $e \in \vec{E}$ till the end of step t, $d^{(t)}(e) = \sum_{t' \in [0,t)} a^{(t')}(e)$. For a node $v \in V$, let $d^{(t)}(v) = \min_{w\in\Gamma(v)}\{d^{(t)}(v \to w)\}$ be the number of fully completed rotations of the rotor at node v at the end of step t. We note that for any arc $u \to v \in \vec{E}$ [26]:

$$0 \leq d^{(t)}(u \to v) - d^{(t)}(u) \leq 1. \tag{1}$$

We also denote $V_i^{(t)} = \{v \in V : d^{(t)}(v) \leq i\}$ and $E_i^{(t)} = \{e \in \vec{E} : d^{(t)}(e) \leq i\}$. For all of the introduced counters, when more than one strategy is considered, the denotation of the strategy will be provided in the subscript of the counter.

Delayed deployment technique In some of the proofs, we will make use of modified executions of the k-agent fair strategies, called *delayed deployments* [16], in which some agents may be stopped at a node, skipping their move for some number of time steps. Formally, a delayed deployment $\mathcal{D}$ of a k-agent fair strategy $\mathcal{U}$ is defined as a function $\mathcal{D} : V \times \mathbb{N} \to \mathbb{N}$, where $0 \leq \mathcal{D}(v,t) \leq k$ is the number of agents which are stopped in node v in step t of the execution of the system. We denote by $delay(\mathcal{U})$ the set of all delayed deployments of strategy $\mathcal{U}$, and by a slight abuse of notation refer to delayed fair strategies $\mathcal{D} \in delay(\mathcal{U})$.

Delayed deployments may be conveniently viewed as algorithmic procedures for delaying agents, and are introduced for purposes of analysis, only. This technique is captured by the following *slow-down lemma*, shown in [16] for the rotor-router model. The slow-down lemma naturally generalizes to all fair strategies; for completeness, we provide a proof of this generalized statement in Appendix A.

Lemma 2.1 *(Slow-down lemma). Let $\mathcal{U}$ be a fair strategy and let $\mathcal{D} \in \text{delay}(\mathcal{U})$. Suppose that the delayed strategy $\mathcal{D}$ covers all the nodes of the graph after T time steps, and in at least τ of these steps, all k agents were active in $\mathcal{D}$. Then, the cover time t_C of undelayed strategy $\mathcal{U}$ can be bounded by:* $\tau \leq t_C \leq T$.

3. Upper bound on cover time

In this section, we will show that any k-agent fair strategy explores any graph in $O(mD/\log k)$ steps, regardless of initialization. We start by providing an informal intuition of the main idea of the proof. After some initialization phase of duration t_0, but before exploration is completed at time t_C, we consider a shortest path connecting the arc of the graph which has already been visited many times at time t_0, with an arc which will remain unvisited at time t_0. We look at the number of visits to consecutive arcs on this path. It turns out that any fair strategy admits a property which can be informally stated as follows: if, up to some step t of exploration, an arc e_{l+1} of the considered path has been traversed more times than the next arc e_l on the path by some difference of δ, then in the next step $t+1$ of exploration, at least $\delta - O(1)$ agents will traverse arcs which have, so far, been visited not more often (up to a constant additive factor) than the arc e_l. In this way, the larger the discrepancy between the number of visits to adjacent arcs, the more activity will the fair strategy perform to even out this discrepancy, by traversing under-visited arcs. This load-balancing behavior of the system will be shown to account for the $\Omega(\log k)$-speedup in cover time with respect to the case of a single agent.

We start by proving two structural lemmas which generalize the results of Yanovski et al. [26, Theorem 2]. The first lemma establishes a connection between the existence of an arc entering a subset of nodes $S \subseteq V$ that has been traversed more times than all arcs outgoing from S, and the number of agents currently located within set S.

Lemma 3.1. *For any (possibly delayed) fair strategy and for any time $t \in \mathbb{N}$ and $d \in \mathbb{N}$, consider the partition of the set of nodes $V = S \cup T$ such that each node in set S sent at most d agents to at least one of its outgoing arcs and each node in set T sent more than d agents to each of its outgoing arcs, i.e., $S = V_d^{(t)}$ and $T = V \setminus S$. Suppose that for some two nodes $v \in S$, $u \in T$, and some $\delta \in \mathbb{N}_+$, we have $d^{(t)}(u \to v) \geq d + \delta$. Then, at least $\delta - 1$ agents are located at nodes from set S at the beginning of step $t+1$.*

Proof. Denote by $S \to T$ (resp., $T \to S$) the set of arcs connecting nodes from S with nodes from T (resp., nodes from T with nodes from S), and let $l = |S \to T| = |T \to S|$. By the basic property of a fair strategy, all arcs outgoing from any node w have been traversed either $d^{(t)}(w)$ or $d^{(t)}(w)+1$ times by the end of step t. From the definition of sets S and T it follows that any arc outgoing from S was traversed at most $d+1$ times and any arc outgoing from T was traversed at least $d+1$ times. By assumption, the arc $u \to v \in T \to S$ was traversed $d + \delta$ times. Hence:

$$\sum_{e \in S \to T} d^{(t)}(e) \leq l \cdot (d+1),$$

$$\sum_{e \in T \to S} d^{(t)}(e) \geq (l-1) \cdot (d+1) + d + \delta \geq \sum_{e \in S \to T} d^{(t)}(e) + \delta - 1.$$

Thus, at least $\delta - 1$ more agents moved from T to S than in the opposite direction until the end of step t. So, at the end of time step t, we have at least $\delta - 1$ agents located at nodes from set S. □

By an application of the above lemma, we obtain the key property of a pair of consecutive arcs which have a different number of traversals at time t.

Lemma 3.2. *For any undelayed fair strategy and for any undirected graph $G = (V, E)$ let $e_2 = u \to v$, $e_1 = v \to w$ be two consecutive arcs of $\vec{G}$. Fix a time step $t \in \mathbb{N}_+$. Then, for any $x \geq d^{(t)}(e_1) + 1$, the number of agents that traverse arcs from set $E_x^{(t)}$ in time step $t+1$ satisfies:*

$$\sum_{e \in E_x^{(t)}} a^{(t)}(e) \geq d^{(t)}(e_2) - d^{(t)}(e_1) - 1.$$

Proof. We can assume that $d^{(t)}(e_2) - d^{(t)}(e_1) \geq 2$, otherwise the claim is trivial. By equation (1), we know that $0 \leq d^{(t)}(e_1) - d^{(t)}(v) \leq 1$ and $d^{(t)}(u) \geq d^{(t)}(e_2) - 1 \geq d^{(t)}(e_1) + 1 > d^{(t)}(v)$. We now apply Lemma 3.1 for $d = d^{(t)}(v)$, putting $S = V_d^{(t)}$ and $T = V \setminus S$. Note that $v \in S$, $u \in T$, and $d^{(t)}(u \to v) = d + \delta$ for $\delta = d^{(t)}(e_2) - d \geq d^{(t)}(e_2) - d^{(t)}(e_1)$. It follows from Lemma 3.1 that during step $t+1$, at least $d^{(t)}(e_2) - d^{(t)}(e_1) - 1$ agents traverse arcs outgoing from nodes from the set S. Since $S = V_d^{(t)}$, all arcs e^* outgoing from nodes from set S have a number of traversals which satisfies $d^{(t)}(e^*) \leq d + 1 \leq d^{(t)}(e_1) + 1$, so $e^* \in E_{d^{(t)}(e_1)+1}^{(t)}$. Thus, $d^{(t)}(e_2) - d^{(t)}(e_1) - 1$ agents in step $t+1$ traverse edges in $E_{d^{(t)}(e_1)+1}^{(t)}$, and moreover $E_{d^{(t)}(e_1)+1}^{(t)} \subseteq E_x^{(t)}$ for all $x \geq d^{(t)}(e_1) + 1$. □

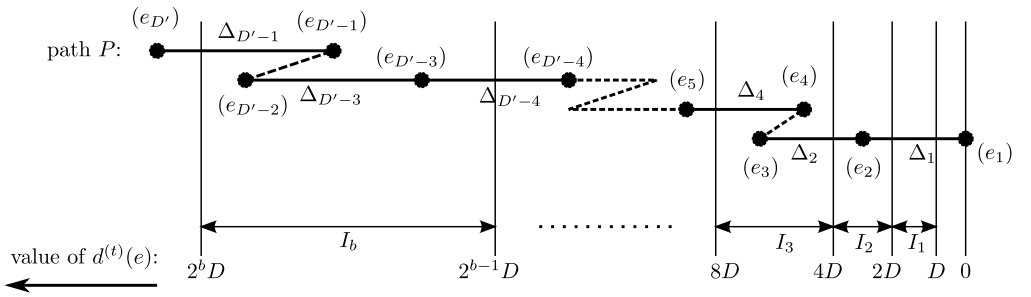


Fig. 1. An illustration of sets I_i and Δ_l in the proof of Theorem 3.3. Dots represent arcs of the graph and their position represent the number of traversals of the corresponding arcs.

The property of fair strategies captured by the above lemma is, in fact, sufficient to prove the main results of this section, following the general approach outlined at the beginning of the section. To show a bound of $t_C \in O(mD/\log k)$, we will apply two separate arguments, first one for the range of relative small k ($k \in 2^{O(D)}$, which corresponds to $t_C \in \Omega(m)$), and then one for values of k which are larger, but polynomially bounded with respect to n.

Theorem 3.3. *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. If $k \leq 2^{16D}$, then a team of k agents performing in parallel any fair strategy, in particular the rotor-router movement, explores G in less than $500mD/\log k$ steps, regardless of the initial positions of agents.*

Proof. First, assume that $k > 2^{160}$ and fix $b = \lfloor (\log k)/2 \rfloor$. Consider the first t_0 steps, where $t_0 = \lceil 2^{b+1} mD/k \rceil$. Since in every step there are exactly k arc traversals, the total number of them during the first t_0 steps is at least $2^{b+1}mD$. We have $2m$ arcs in total. Thus, there exists an arc e' such that $d^{(t_0)}(e') \geq 2^b D$. These first t_0 steps we will call as a form of setup stage, after which we begin to analyze the behavior of the process.

Denote by t_C the cover time of G with k agents for a given initialization. We will assume that $t_C > t_0$, i.e., at least one arc of the graph has not been explored at time t_0; otherwise, $t_C \leq t_0 = \lceil 2^{b+1} mD/k \rceil \leq \lceil 2mD/\sqrt{k} \rceil$, since $b = \lfloor (\log k)/2 \rfloor$, and the claim of the theorem holds for all k.

Take $e'' \in \vec{E}$ to be an arc which is explored for the first time in step t_C, i.e., such that $d^{(t_C-1)}(e'') = 0$. Since the diameter of G is D, there exists a path $\mathcal{P} = \langle e'' = e_1, e_2, \ldots, e_{D'} = e' \rangle$ such that $D' \leq D + 2$, and for each $l \in [1, D']$, $e_l = v_{l+1} \to v_l$ where $v_l, v_{l+1} \in V$.

Fix a time step $t \in [t_0, t_C)$. We will place some of the arcs of path $\mathcal{P}$ in groups (buckets) $I_1, I_2, \ldots, I_b$, such that all arcs in bucket I_i have been traversed between $2^{i-1}D$ and $2^i D$ times until step t. Formally, denote:

$$I_i = \left\{ l : d^{(t)}(e_l) \in [2^{i-1}D, 2^i D) \right\} \subseteq [1, D'], \quad \text{for } i \in [1, b].$$

We now analyze to which buckets the successive arcs of the path $\mathcal{P}$ belong. For $l \in [1, D')$, define

$$\Delta_l = \begin{cases} [d^{(t)}(e_l), d^{(t)}(e_{l+1})), & \text{if } d^{(t)}(e_l) < d^{(t)}(e_{l+1}), \\ \emptyset, & \text{otherwise.} \end{cases}$$

Note that the union of all Δ_l covers the interval $[0, 2^b D)$, since for any $x \in [0, 2^b D)$ there exists $l^* \in [1, D')$ such that $x \in \Delta_{l^*}$ because $d^{(t)}(e_1) = 0$ and $d^{(t)}(e_{D'}) \geq 2^b D$ (see Fig. 1 for an illustration). The intuition of the proof is now as follows: Since there are at most D' non-empty intervals Δ_l spanning the total range $[0, 2^b D)$ of all buckets $I_1, I_2, \ldots, I_b$, in a large number (linear in b) of these buckets I_i, the average length of an interval Δ_l starting in bucket I_i will be at least $|I_i| b/D = 2^{i-1}b$, up to a constant factor. The existence of such long intervals Δ_l beginning in I_i will allow us to exploit Lemma 3.2 to show that arcs e_l, e_{l+1} differ in the number of traversals by a constant times $2^{i-1}b$. This implies that for the considered bucket indices i, the number of agents active at time t on edges from buckets $I_1, \ldots, I_i$ will be at least $2^{i-1}b$, up to constant factors and minor shifts at bucket boundaries. We now proceed to formalize the above arguments.

For $i \in [1, b]$, denote by $\mathcal{X}_i$ the set of intervals Δ_l beginning in bucket I_i: $\mathcal{X}_i = \bigcup_{l \in I_i} \Delta_l$. Consider any $x \in [0, 2^b D)$, and let l^* be such that $x \in \Delta_{l^*}$. We have $d^{(t)}(e_{l^*}) \leq x < 2^b D$, hence $l^* \in I_{i^*}$, for some $i^* \in [1, b]$, and $x \in \mathcal{X}_{i^*}$. It follows that:

$$[0, 2^b D) \subseteq \bigcup_{i \in [1,b]} \mathcal{X}_i. \tag{2}$$

For $i \in \mathbb{N}$, denote by $a_i^{(t)}$ the number of agents that traverse arcs from set $E^{(t)}_{2^i D}$ in step $t+1$, $a_i^{(t)} = \sum_{e \in E^{(t)}_{2^i D}} a^{(t)}(e)$, and let $a_{-1}^{(t)} = 0$. (We remark that $E^{(t)}_{2^i D} \supseteq \{e_j : j \in I_1 \cup \ldots \cup I_i\}$.) First, note that for all $i \in [1, b]$ and for $l \in I_i$, we have $d^{(t)}(e_l) < 2^i D$. So, by Lemma 3.2:

$$a_i^{(t)} \geq d^{(t)}(e_{l+1}) - d^{(t)}(e_l) - 1 = |\Delta_l| - 1 \quad \Longrightarrow \quad |\Delta_l| \leq a_i^{(t)} + 1. \tag{3}$$

Now, observe that for any $i \in [1, b]$:

$$\max \mathcal{X}_i = \max_{l \in I_i} (\max \Delta_l) \leq \max_{l \in I_i} \left(d^{(t)}(e_l) + |\Delta_l| - 1 \right) < 2^i D + a_i^{(t)}, \tag{4}$$

where we took into account inequality (3) and that $d^{(t)}(e_l) < 2^i D$ for $l \in I_i$.

Next, we will show that for all $i \in [1, b]$:

$$2^{i-1} D - a_{i-1}^{(t)} \leq |\mathcal{X}_i| \leq |I_i|(a_i^{(t)} + 1). \tag{5}$$

The right inequality in (5) is proved as follows: $|\mathcal{X}_i| \leq \sum_{l \in I_i} |\Delta_l| \leq |I_i|(a_i^{(t)} + 1)$, where the latter inequality is a consequence of (3).

We now prove the left inequality in (5). If $a_{i-1}^{(t)} \geq 2^{i-1} D$, then the bound is trivial. In the case when $a_{i-1}^{(t)} < 2^{i-1} D$, we will first prove that:

$$[2^{i-1} D + a_{i-1}^{(t)}, 2^i D) \subseteq \mathcal{X}_i. \tag{6}$$

To this end, take any $x \in [2^{i-1} D + a_{i-1}^{(t)}, 2^i D)$ and observe that by (2), there exists some $j \in [1, b]$ such that $x \in \mathcal{X}_j$. Moreover, note that:

1. For any $j < i$, $x \notin \mathcal{X}_j$, because, by (4), $\max \mathcal{X}_j < 2^j D + a_j^{(t)} \leq 2^{i-1} D + a_{i-1}^{(t)} \leq x$.
2. For any $j > i$, $x \notin \mathcal{X}_j$, because: $\min \mathcal{X}_j = \min_{l \in I_j, \Delta_l \neq \emptyset} \min \Delta_l = \min_{l \in I_j, \Delta_l \neq \emptyset} d^{(t)}(e_l) \geq 2^{j-1} D \geq 2^i D > x$.

Thus, $x \in \mathcal{X}_i$, and (6) follows. Equation (6) implies that

$$|\mathcal{X}_i| \geq 2^i D - \left(2^{i-1} + a_{i-1}^{(t)} \right) = 2^{i-1} D - a_{i-1}^{(t)},$$

which completes the proof of (5). Next, by (5),

$$|I_i| \geq \frac{2^{i-1} D - a_{i-1}^{(t)}}{a_i^{(t)} + 1} \quad \text{for all } i \in [1, b].$$

The buckets $I_1, I_2, \ldots, I_b$ are pairwise disjoint by definition and contain at most D' elements altogether, which gives:

$$D + 2 \geq D' \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1} D - a_{i-1}^{(t)}}{a_i^{(t)} + 1} \geq \sum_{i=1}^{b} \frac{2^{i-1} D}{a_i^{(t)} + 1} - b,$$

where in the last inequality we used the fact that $a_i^{(t)} \geq a_{i-1}^{(t)}$ for $i \in [2, b]$. Dividing the sum in the last inequality by bD, we get the following expression for the arithmetic average of values $\frac{2^{i-1}}{a_i^{(t)}+1}$:

$$\frac{1}{b} \sum_{i=1}^{b} \frac{2^{i-1}}{a_i^{(t)} + 1} \leq \frac{D + b + 2}{bD} = \frac{1}{b} + \frac{1 + 2/b}{D} < \frac{9.2}{b},$$

where in the last inequality we took into account that $k \leq 2^{16D}$ and $b \leq (\log k)/2$ by assumption, hence $D \geq (\log k)/16 \geq b/8$, and that $b = \lfloor (\log k)/2 \rfloor \geq 80$. All the elements of the considered sum are positive, hence by Markov's inequality, there exists a subset of indices $S^{(t)} \subseteq [1, b]$, with $|S^{(t)}| \geq b/2$, such that for all $j \in S^{(t)}$, value $\frac{2^{j-1}}{a_j^{(t)}+1}$ is at most twice the arithmetic average:

$$\frac{2^{j-1}}{a_j^{(t)} + 1} \leq 2 \cdot \frac{1}{b} \sum_{i=1}^{b} \frac{2^{i-1}}{a_i^{(t)} + 1} < \frac{18.4}{b}.$$

This implies that for all $j \in S^{(t)}$:

$$a_j^{(t)} \geq \tfrac{b}{18.4} \cdot 2^{j-1} - 1 > \tfrac{b}{25} \cdot 2^{j-1}, \tag{7}$$

where we again took into account that $b \geq 80$. Observe that we showed equation (7) for any $t \in [t_0, t_C)$.

We are ready to complete the proof of the theorem. Fix $t_1 = \lceil 100mD/b \rceil$. We now prove that

$$t_C \leq t_0 + 2t_1 + 4m. \tag{8}$$

Suppose, by contradiction, that $t_C > t_0 + 2t_1 + 4m$. We will say that an index $j \in [1,b]$ is *good* after time t if $j \in S^{(t)}$. Since for all $t \in [t_0, t_C)$ we have $|S^{(t)}| \geq b/2$ and $S^{(t)} \subseteq [1,b]$, by the pigeon-hole principle there must exist an index j^* that is good in at least $(t_C - t_0)/2 > t_1 + 2m$ steps in $[t_0, t_C)$; we will call these steps *good steps*.

For an arc e of the graph, we denote by t_e the so called *exit time step* for arc e, after which the total number of traversals of e for the first time exceeds $2^{j^*}D$: $d^{(t_e)}(e) \leq 2^{j^*}D < d^{(t_e+1)}(e)$. The set of all exit time steps, taken over all arcs of the graph, is denoted $\hat{T} = \{t_e : e \in \vec{E}\}$. Note that $e \in E^{(t)}_{2^{j^*}D}$ if and only if $t \leq t_e$, and therefore we may write:

$$\sum_{t\in[0,t_C)\setminus\hat{T}} a^{(t)}_{j^*} = \sum_{t\in[0,t_C)\setminus\hat{T}} \sum_{e\in E^{(t)}_{2^{j^*}D}} a^{(t)}(e) \leq \sum_{e\in\vec{E}} \sum_{t=0}^{t_e-1} a^{(t)}(e) \leq \sum_{e\in\vec{E}} d^{(t_e)}(e) \leq 2m \cdot 2^{j^*} D. \tag{9}$$

Now, recall that there are at least $t_1 + 2m$ good time steps $t \in [t_0, t_C)$ for which index j^* satisfies (7), and that $|\hat{T}| \leq 2m$. It follows that:

$$\sum_{t\in[0,t_C)\setminus\hat{T}} a^{(t)}_{j^*} > t_1 \cdot \frac{b}{25} \cdot 2^{j^*-1} = \left\lceil \frac{100mD}{b} \right\rceil \frac{b}{25} \cdot 2^{j^*-1} \geq 2m \cdot 2^{j^*} D,$$

a contradiction with (9). Thus, we have proved (8). By (8), we obtain

$$t_C \leq t_0 + 2t_1 + 4m = \left\lceil \frac{2^{b+1}mD}{k} \right\rceil + 2\left\lceil \frac{100mD}{b} \right\rceil + 4m \leq$$
$$\leq \frac{mD}{\log k}\left(\frac{2^{b+1}\log k}{k} + \frac{200\log k}{b} + \frac{4\log k}{D} + \frac{3\log k}{mD}\right). \tag{10}$$

Taking into account that $b = \lfloor (\log k)/2 \rfloor$, $k \leq 2^{16D}$, and $k > 2^{160}$, we obtain that the expression in the above bracket can be bounded by a constant, giving: $t_C < 500\frac{mD}{\log k}$. This completes the proof for the case $k > 2^{160}$.

Suppose now that $k \leq 2^{160}$. Yanovski et al. [26] showed that a single agent explores the graph in at most $2mD$ steps regardless of the initialization, and moreover, that adding agents cannot decrease the number of traversals on any edge. We thus trivially obtain the claim: $t_C \leq 2mD < 500\frac{mD}{\log k}$. □

Now our goal is to extend the previous result for the case of $k \geq 2^{16D}$. In the following lemma we will require the property that at most one agent traverses an arc in a single step. In order to obtain this property in the rotor-router, it is sufficient to obtain a state in which each agent occupies a distinct vertex. It is possible to prove that from such configuration on, no arc is traversed by two agents simultaneously. However, for general fair strategies this may not be enough, as the strategy might choose to select the same outgoing arc twice in a row. To deal with this we introduce a class of specific delayed fair strategies called *deferred fair strategies*.

Definition 3.1. Let $\mathcal{U}^*$ be a delayed deployment and let v be any node. For each time step t, let $a_t \in \mathbb{N}_+$ and $0 \leq b_t < \deg(v)$ be such that the total number of traversals of arcs outgoing from v by the beginning of step t is $a_t \cdot \deg(v) + b_t$. We say that $\mathcal{U}^*$ is *deferred* if for each time step t the following conditions hold:

(i) if v contains at most $\deg(v) - b_t$ agents, then all those agents are propagated in step t,
(ii) if v contains more than $\deg(v) - b_t$ agents, then $\deg(v) - b_t$ agents are propagated in step t and all the remaining agents are blocked at v during round t.

In the following lemma we show that if each agent starts from a distinct node, then deferred strategies propagate agents efficiently, with only a constant factor delay with respect to an undelayed strategy, and moreover that in such a deffered strategy no two agents ever traverse the same arc simultaneously. Denote by $\mathsf{out}(v,t)$ ($\mathsf{in}(v,t)$) the total number of traversals of arcs outgoing from (incoming to) v until the beginning of step t. By $\mathsf{init}(v)$ we denote the number of agents initially located at v.

Lemma 3.4. *For any deferred fair strategy starting from a configuration with each agent located at a distinct node:*

1. *each arc is traversed at most once in a time step,*
2. *at least k arc traversals are performed by agents in any two consecutive time steps in total,*
3. *for any vertex v and time t, we have* $\mathsf{out}(v, t+2) \geq \mathsf{in}(v,t) + \mathsf{init}(v)$.

Proof. We will first show Claim 1. Consider a node v at the beginning of any time step t. Note that from the definition of deferred strategies, from among the arcs outgoing from v only arcs with the same number of traversals are traversed in

step t. Moreover, at most $\deg(v)$ agents are sent from v in time step t. Thus, in round t any arc outgoing from v is traversed at most once.

To prove 2, we will show the following fact by induction on t: at the beginning of step t, any vertex v is in one of the two following states, depending on whether $\deg(v)$ divides $\mathrm{out}(v,t)$, or not:

(∗) $\deg(v) \nmid \mathrm{out}(v,t)$ and at the beginning of step t at node v there are no agents that were delayed from previous rounds,
(∗∗) $\deg(v) \mid \mathrm{out}(v,t)$ and in v at the beginning of step t there are at most $\deg(v)-1$ agents that were delayed from previous rounds.

At the beginning of steps 1 and 2 each node is in one of the above two states because each node initially contains no more than one agent and so no agent is delayed during round 1. Assume that all nodes are in one of the states (∗) or (∗∗) at the beginning of step $t \geq 2$. Since each arc is traversed at most once in step $t-1$, then x, the number of agents entering v in step t satisfies $x \leq \deg(v)$. If v is in state (∗∗) at the beginning of step t, then it contains $z < \deg(v)$ delayed agents at the beginning of step t. If $x+z > \deg(v)$, then v during round t propagates $\deg(v)$ agents and the node remains in state (∗∗) having $x+z-\deg(v) < \deg(v)$ delayed agents. Otherwise, it propagates all its agents and changes its state to (∗), unless $x+z = \deg(v)$ in which case it remains in state (∗∗). Thus, at the beginning of step $t+1$, the node v is in one of the states (∗) or (∗∗). Assume that v is in state (∗) at the beginning of step t. Let r be the remainder modulo $\deg(v)$ of $\mathrm{out}(v,t)$. If $x+r < \deg(v)$, then all x agents are propagated during step t and the node remains in state (∗). Otherwise, $x+r-\deg(v) < \deg(v)$ agents are delayed and the node changes its state to (∗∗). Thus, in this case also at the beginning of $t+1$, node v is in one of the states (∗) or (∗∗).

Consider now the total number of edge traversals in steps t and $t+1$. Observe that all agents that did not make a move during round t are located at nodes that are in state (∗∗) at the beginning of round $t+1$. A node in state (∗∗) containing z delayed agents propagates at least z agents during round $t+1$ thus in total during rounds t and $t+1$ the total number of traversals is at least k. This also shows 3, because the total number of exits from v until the beginning of step $t+2$ equals at least the total number of visits to v until the beginning of step t. □

This observation allows us to show the counterpart of Lemma 3.2 for deferred strategies.

Lemma 3.5. *Let $G=(V,E)$ be any undirected graph, let $\mathcal{U}^*$ be any deferred strategy with each agent starting from a distinct node and let $e_2 = u \to v$, $e_1 = v \to w$ be two consecutive arcs of $\vec{G}$. Fix a time step $t \in \mathbb{N}_+$. Then, for any $x \geq d^{(t)}(e_1)+2$, the total number of agents following $\mathcal{U}^*$, that traverse arcs from set $E_x^{(t)}$ in time steps $t+1$ and $t+2$ satisfies:*

$$\sum_{e \in E_x^{(t)}} (a^{(t)}(e) + a^{(t+1)}(e)) \geq d^{(t)}(e_2) - d^{(t)}(e_1) - 1.$$

Proof. We can assume that $d^{(t)}(e_2) - d^{(t)}(e_1) \geq 2$, otherwise the claim is trivial. By equation (1), we know that $0 \leq d^{(t)}(e_1) - d^{(t)}(v) \leq 1$ and $d^{(t)}(u) \geq d^{(t)}(e_2) - 1 \geq d^{(t)}(e_1) + 1 > d^{(t)}(v)$. We now apply Lemma 3.1 for $\mathcal{U}^*$ and $d = d^{(t)}(v)$, putting $S = V_d^{(t)}$ and $T = V \setminus S$. Note that $v \in S$, $u \in T$, and $d^{(t)}(u \to v) = d + \delta$ for $\delta = d^{(t)}(e_2) - d \geq d^{(t)}(e_2) - d^{(t)}(e_1)$. It follows from the lemma that at the beginning of step $t+1$, at least $d^{(t)}(e_2) - d^{(t)}(e_1) - 1$ agents are located in the nodes from the set S. By Lemma 3.4, each agent traverses an arc during steps $t+1$ and $t+2$, in particular, each agent located at a node in S traverses and arc. Consequently, at least $d^{(t)}(e_2) - d^{(t)}(e_1) - 1$ agents will traverse arcs outgoing from nodes from S in steps $t+1$ and $t+2$.

Consider the number of traversals of arcs outgoing from S in at the beginning of steps $t+1$ and $t+2$. Since $S = V_d^{(t)}$, each arc e^* outgoing from a node in the set S has at the end of step t the number of traversals which satisfies $d^{(t)}(e^*) \leq d+1 \leq d^{(t)}(e_1)+1$. Thus,

$$e^* \in E^{(t)}_{d^{(t)}(e_1)+1} \subseteq E^{(t)}_{d^{(t)}(e_1)+2}.$$

Thus, in total at least $d^{(t)}(e_2) - d^{(t)}(e_1) - 1$ agents in steps $t+1$ and $t+2$ traverse arcs in $E^{(t)}_{d^{(t)}(e_1)+2}$, and moreover $E^{(t)}_{d^{(t)}(e_1)+2} \subseteq E_x^{(t)}$ for all $x \geq d^{(t)}(e_1)+2$. □

To obtain our claim about the speedup of fair strategies, we first make an additional assumption that each agent starts from a distinct node. We showed that for deferred strategies this additional assumption implies that no arc is traversed by more than one agent in a single step. The proof then proceeds along similar lines as that of Theorem 3.3, and we show that in many time steps t, there exists a pair of arcs e_{l+1}, e_l in $\mathcal{P}$ with a large difference in the number of traversals up to time t. However, instead of counting the number of long arcs on path $\mathcal{P}$ belonging to a bucket I_i, in this proof we take advantage of the fact that the length of the path $D' \leq D+2$ is small compared to $\log k$, which can be used to infer the existence of the sought arc pairs.

Lemma 3.6. *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. If $k \geq 2^{16D}$, then a team of k agents performing in parallel any fair strategy, with each agent starting from a distinct node of the graph, explores G in time $54mD/\log k$.*

Proof. Take the considered fair strategy $\mathcal{U}$ and construct its deferred counterpart $\mathcal{U}^*$. We will show the lemma for $\mathcal{U}^*$ and the lemma for $\mathcal{U}$ will follow from Lemma 2.1 since $\mathcal{U}^* \in delay(\mathcal{U})$.

Denote by t_C the cover time of graph G. Let

$$X = \left\lfloor k^{1/(2D+6)} \right\rfloor \quad \text{and} \quad Y_i = 2 + \sum_{j=0}^{i-1} X^j = 2 + \frac{X^i - 1}{X - 1} \quad \text{for } i \in \mathbb{N}_+.$$

Note that since $k \geq 2^{16D}$, we have:

$$X \geq 2 \quad \text{and} \quad Y_i < 2 + X^i \text{ for all } i \in \mathbb{N}_+. \tag{11}$$

Similarly as in proof of Theorem 3.3, we first consider a setup phase, consisting of steps $[1, t_0)$ of exploration, this time defining t_0 as:

$$t_0 = 2\left\lceil \frac{m(X^{2D+5}+2)}{k} \right\rceil \leq 2\left\lceil m\left(\frac{1}{X} + \frac{2}{k}\right)\right\rceil. \tag{12}$$

During the setup stage, the total number of edge traversals is at least $2m(X^{2D+5}+2)$ since by Lemma 3.4 in every two consecutive steps, the agents are making at least k arc traversals. Thus, there exists an arc e' such that $d^{(t_0)}(e') \geq X^{2D+5}+2$. There also exists an arc e'' such that $d^{(t_C-1)}(e'') = 0$. Thus, for each $t \in [t_0, t_C)$,

$$d^{(t)}(e'') = 0 \text{ and } d^{(t)}(e') \geq X^{2D+5} + 2 > Y_{2D+5}. \tag{13}$$

Since D is the diameter of G, there exists a path $\mathcal{P} = \langle e'' = e_1, e_2, \ldots, e_{D'} = e' \rangle$, such that $D' \leq D+2$ and for all $i \in [1, D')$, $e_i = v_i \to v_{i+1}$ where $v_i, v_{i+1} \in V$.

For each time step t and $i \geq 2$, let $a_i^{(t)}$ be the number of agents that during step $t+1$ traverse those arcs which were traversed at most Y_i times until the end of step t, $a_i^{(t)} \equiv \sum_{e \in E_{Y_i}^{(t)}} a^{(t)}(e)$. We have for any $i \geq 2$:

$$\sum_{t=t_0}^{t_C-1} a_i^{(t)} \leq 2m(Y_i+1) < 3mY_i, \tag{14}$$

and we prove the first inequality by contradiction. Thus, if the first inequality does not hold, then some arc e contributes at least $Y_i + 2$ to the above sum. Then, since in each time step $t \in \mathbb{N}$ each arc is traversed at most once (by Lemma 3.4), there exist Y_i+2 steps $t_1, \ldots, t_{Y_i+2}$, where $t_0 < t_1 < t_2 < \cdots < t_{Y_i+2} \leq t_C$, in which e is traversed, and moreover $e \in E_{Y_i}^{(t_{Y_i+2}-1)}$ by definition of $a_i^{(t)}$. However, till the end of step $t_{Y_i+2} - 1 \geq t_{Y_i+1}$ the arc e has been traversed $Y_i + 1$ times, so, $e \notin E_{Y_i}^{(t_{Y_i+2}-1)}$, and we obtain a contradiction, proving (14).

We now prove that

$$t_C \leq t_0 + 12\left\lceil \frac{m}{X-1} \right\rceil. \tag{15}$$

Suppose, by contradiction, that $t_C > t_0 + 12\lceil m/(X-1)\rceil$.

For each time step t, we will call the set of arcs $E_{Y_{i+1}}^{(t)} \setminus E_{Y_i}^{(t)}$ the *i-th zone* at time t, for $i \geq 2$.

Each zone that does not contain any arc of path $\mathcal{P}$ in a given time step is called *free*. The path $\mathcal{P}$ has D' arcs and hence at least D' zones with indices in the interval $[2, 2D'+1]$ are free in each time step. Thus, by the pigeonhole principle, during the time period $[t_0, t_C)$ there must exist an index $i^* \in [2, 2D'+1]$ such that the i^*-th zone is free during a set of time steps $T \subseteq [t_0, t_C)$, with:

$$|T| \geq (t_C - t_0)/2 > 6\lceil m/(X-1)\rceil.$$

By (13), the arc e' belongs to a zone with index at least $2D+6 \geq 2D'+2$ in each time step $t \in T$, while arc e'' belongs to zone 1. Since the i^*-th zone is free at each time $t \in T$, by following path $\mathcal{P}$ from arc e' to e'', we will necessarily encounter an index $j \in [1, D')$, such that $d^{(t)}(e_{j+1}) \geq Y_{i^*+1} + 1$ and $d^{(t)}(e_j) \leq Y_{i^*}$, which gives:

$$d^{(t)}(e_{j+1}) - d^{(t)}(e_j) \geq Y_{i^*+1} + 1 - Y_{i^*} = X^{i^*} + 1.$$

By Lemma 3.5, for each $t \in T$, at least X^{i^*} agents traverse arcs from set $E_{Y_{i^*}}^{(t)}$ in steps $t+1$ and $t+2$, i.e., $(a_{i^*}^{(t)} + a_{i^*}^{(t+1)}) \geq X^{i^*}$. Thus,

$$\sum_{t\in T}(a_{i^*}^{(t)}+a_{i^*}^{(t+1)}) \geq |T|X^{i^*} \geq 6\left\lceil \frac{m}{X-1}\right\rceil X^{i^*} > 6mY_{i^*},$$

and we obtain

$$\sum_{t=t_0}^{t_C-1} a_i^{(t)} \geq \frac{1}{2}\sum_{t\in T}(a_{i^*}^{(t)}+a_{i^*}^{(t+1)}) \geq 3mY_{i^*}.$$

This contradicts (14), completing the proof of (15).

Recall that $X=\lfloor k^{1/(2D+6)}\rfloor$. By (15), (12), and the definition of X, we have:

$$t_C \leq 2\left\lceil m\left(\frac{1}{X}+\frac{2}{k}\right)\right\rceil + 12\left\lceil \frac{m}{X-1}\right\rceil \leq 14\frac{m}{X-1}+\frac{4m}{k}+14 \leq \frac{mD}{\log k}\left(\frac{\log k}{D(k^{1/(8D)}-2)}+\frac{4\log k}{Dk}+14\right).$$

Observe that for fixed D, the expression in the above bracket is strictly decreasing with k for $k>2^{8D}$, and for $k=2^{16D}$ takes a value of 54. Knowing that $k\geq 2^{16D}$, we therefore obtain $t_C \leq 54\frac{mD}{\log k}$. □

It remains to consider the case not covered by the above lemma, when not all agents start from distinct positions. In fact, we will reduce such a case to the one already considered by making use of the concept of delayed deployments discussed in Section 2.

Lemma 3.7. *Let $\mathcal{R}$ and $\mathcal{R}'$ be two k-agent fair strategies with cover times t_C and t'_C, respectively. Suppose that there exists a delayed deployment $\mathcal{D}\in delay(\mathcal{R})$ whose execution transforms the starting configuration of $\mathcal{R}$ into the starting configuration of $\mathcal{R}'$ in $\hat{t}$ time steps. Then, $t_C \leq \hat{t}+t'_C$.*

Proof. Observe that the concatenation of the execution of deployment $\mathcal{D}$ for $\hat{t}$ steps and $\mathcal{R}'$ for t'_C steps is a delayed deployment of $\mathcal{R}$ which explores the graph in $t_C\leq \hat{t}+t'_C$ steps. The claim follows by Lemma 2.1. □

The next lemma provides an upper bound on the time of transforming any configuration of a fair strategy with $k\leq n$ agents into one in which agents occupy distinct starting nodes.

Lemma 3.8. *For any initialization $\mathcal{R}$ of a fair strategy with k agents, $k\leq n$, there exists a delayed fair strategy $\mathcal{D}\in delay(\mathcal{R})$ which terminates in a configuration in which all agents occupy distinct positions after $\hat{t}\leq k^4$ steps.*

Proof. In deployment $\mathcal{D}$, we release agents sequentially from their starting positions in $\mathcal{R}$, moving one agent only at a time until it is located at a node unoccupied by another agent. Consider the phase in which we move a fixed agent a in this deployment. In the worst case, a has to explore the graph induced by all nodes occupied to date. The agent acts a single-agent rotor router system with respect to this graph. Recall that the cover time of a graph with m edges and diameter D by a single agent is at most $2mD$, regardless of the initial configuration [26]. Since in the considered system there are at most k occupied nodes with at most $k^2/2$ edges between them, and the graph of occupied nodes has diameter at most k, a finds an unoccupied node within $2\cdot k^2/2\cdot k=k^3$ steps. This has to be done by each of k agents, thus the total time of all phases of the delayed deployment is $\hat{t}\leq k^4$. □

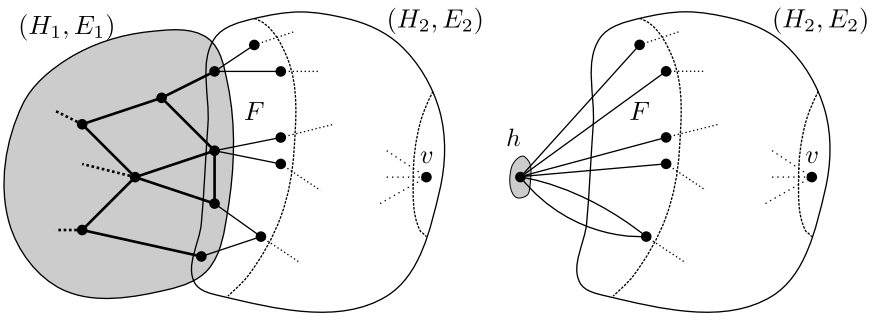
When $1<k\leq\lceil n^{1/5}\rceil$, we can bound the time $\hat{t}$ in the above lemma as:

$$\hat{t}\leq k^4\leq \lceil n^{1/5}\rceil^4 \leq (n^{1/5}+1)^4 = n^{4/5}+4n^{3/5}+6n^{2/5}+4n^{1/5}+1\leq 16n^{4/5}\leq 16n/k\leq 32m/\log k.$$

Combining the above result with Lemmas 3.6 and 3.7, we obtain that for any fair strategy with arbitrary initialization with k agents, $k\leq\lceil n^{1/5}\rceil$ and $k\geq 2^{16D}$, exploration is completed within time $t_C=\hat{t}+t'_C\leq 32\frac{mD}{\log k}+54\frac{mD}{\log k}=86\frac{mD}{\log k}$. On the other hand, when $k<2^{16D}$, by Theorem 3.3, the cover time is $t_C\leq 500\frac{mD}{\log k}$. It follows that the bound $t_C\leq 500\frac{mD}{\log k}$ holds for all starting configurations with $k\leq\lceil n^{1/5}\rceil$.

When $k>\lceil n^{1/5}\rceil$, we can make use of a result of Yanovski et al. [26] (generalized in this paper to fair strategies), stating that the worst-case initialization of a fair strategy system with k agents cannot have greater cover time than the worst-case initialization of a system with $k'<k$ agents. Putting $k'=\lceil n^{1/5}\rceil$, for any $k>\lceil n^{1/5}\rceil$ we obtain: $t_C\leq 500\frac{mD}{\log k'}\leq 2500\frac{mD}{\log n}$. Finally, combining the results for $k\leq\lceil n^{1/5}\rceil$ and $k>\lceil n^{1/5}\rceil$ gives the following theorem.

Theorem 3.9. *Let $G=(V,E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. A team of k agents performing in parallel any fair strategy, in particular the rotor-router movement, explores G in time $\max\{500mD/\log k, 2500mD/\log n\}$, regardless of the initial positions of agents. In particular, if $k\leq n^c$ for some $c>0$, then the cover time is at most $2500c\cdot mD/\log k$.* □



(a) Graph decomposition into "deep" part H_2 and "heavy" part H_1.

(b) Ilustration of graph G' obtained by contracting the "heavy" part.

Fig. 2. Graph decompositions used in the proof of Theorem 4.1.

Theorems 3.3 and 3.9 imply that the cover time of the rotor-router is $O(mD/\log k)$ for all graphs, whenever $k \in 2^{O(D)}$ or $k \in O(\text{poly}(n))$.

4. Lower bound on cover time

Theorem 4.1. *Let $G = (V, E)$ be any undirected graph of diameter D. There exists a port labeling of the edges of G, an initialization of pointers and an assignment of starting positions to a team of k agents, such that the exploration performed in parallel with the rotor-router movement has cover time $t_C \geq \frac{1}{4}mD/k$.*

Proof. If $k > m$, we make all agents start from an arbitrarily chosen single node, and choose an arbitrary pointer initialization. In such scenario, the exploration will be completed after time at least $D > \frac{mD}{k}$. Thus, we can safely assume that $k \leq m$.

For any graph $G = (V, E)$, as shown in [3, Theorem 2], there exists a partition of the edge set $E = E_1 \cup E_2$, such that:

(i) $|E_1| \geq \frac{m}{2}$,
(ii) there exist $V_1 \subseteq V$ and $V_2 \subseteq V$ such that the subgraphs $H_1 = G[V_1]$ and $H_2 = G[V_2]$ are connected and their edge sets are E_1 and E_2, respectively,
(iii) there exists a node $v \in V_2$ being at distance at least $\frac{D}{2}$ from each node of H_1.

Fig. 2(a) illustrates such a partition. Denote by $F \subset E_2$ the set of edges incident to some node from H_1.

Now, let $C = \{e_1, e_2, \ldots, e_{2|E_1|}\}$ be a directed Eulerian cycle in $\vec{H}_1$ (the bidirected subgraph corresponding to H_1) traversing every edge in E_1 exactly once in each direction. To simplify notation, let $\Delta = \left\lfloor \frac{2|E_1|}{k} \right\rfloor$. We choose an arbitrary set of indexes $1 = j_1 < j_2 < \ldots < j_k \leq 2|E_1|$ such that they are spread (almost-)equidistantly in $\{1, \ldots, 2|E_1|\}$, that is:

$$\forall i \in [1, k) \quad j_{i+1} - j_i \in \{\Delta, \Delta + 1\} \quad \text{and} \quad j_1 - j_k + 2|E_1| \in \{\Delta, \Delta + 1\}.$$

This is possible because, due to (i), $2|E_1| \geq k$.

We partition E_1 into Δ sets $S_1, \ldots, S_\Delta$ of size k:

$$S_{i+1} = \left\{e_{j_1+i}, e_{j_2+i}, \ldots, e_{j_k+i}\right\}, \quad \text{for } 0 \leq i < \Delta,$$

and one set for all remaining edges: $R = E_1 \setminus \bigcup_{t=1}^{\Delta} S_t$. Informally, we divide the Eulerian cycle into segments of roughly the same sizes and then S_1 gets first edge from each segment, S_2 gets the second edge from each segment and so on. As the sizes of segments may differ by one, some segments may have an extra edge and those make up the set R.

We choose the starting positions of k agents, the port assignment, and the initialization of pointers for the edges in E_1 such that in their first $\Delta + 1$ steps, the k agents traverse all edges in E_1 in the following delayed deployment: for each $t \in \{1, \ldots, \Delta\}$, in the t-th step, exactly the edges in S_t are traversed, whereas in the $(\Delta + 1)$-th step we delay some agents so that exactly the edges in R are traversed. We achieve this by setting outgoing ports so that, for every node u in H_1, we order the edges in E_1 incident to u by assigning smaller ports to edges in S_t than to the edges in S_{t+1}, for each $t \in \{1, \ldots, \Delta\}$, where $S_{\Delta+1} = R$. Such a port ordering is enough to explore the graph H_1, with delayed deployment, with the property that every edge is visited once every $\Delta + 1$ steps.

Now we assign ports to the edges in F. To this end, we consider the subgraph of G, denoted by $\widetilde{G}$, consisting of the edges in $E_1 \cup F$. In other words, we take H_1 (together with the port assignment obtained above) and we add the edges in F, obtaining $\widetilde{G}$. Note that, by (ii), each edge in F has one endpoint in V_1 and the other endpoint in $V \setminus V_1$. The ports in F

are determined by analyzing the behavior of agents in the graph $\widetilde{G}$ in the delayed deployment described above. Whenever any set of agents are about to leave H_1 and traverse any edge from F, we select a single agent in a deterministic way (for example, by choosing the agent located on a node with the smallest index, having indexes assigned to nodes). We stop all other agents and perform traversals only with the selected agent, until it returns to H_1. We set the ports of the edges in F so that whenever an agent leaves H_1 through an edge $(v \to u) \in F$ ($v \in V_1, u \notin V_1$), it returns to H_1 through the edge $(u \to v)$ (we call this property *the property of return*). Having the property of return, we achieve that the agents patrol E_1, and whenever an agent is about to leave H_1, the other agents are delayed until the agent returns to the same node. Since the selection of agents is done deterministically, the edges in F are always traversed in separated periods of time (when one agent is traversing edges from F, all other agents are stopped) in a cyclic fashion, i.e., the sequence of traversal of the edges in F is $(f_1, f_1', f_2, f_2', \ldots, f_{|F|}, f_{|F|}')^*$, where f' means the reversed edge to an edge f, i.e., if $f = (u \to v)$, then $f' = (v \to u)$. Denote $f_i = (u_i \to v_i)$ for each $i \in \{1, \ldots, |F|\}$.

It remains to assign port labels to the edges in $E_2 \setminus F$, and to initialize the remaining pointers for the nodes in $V \setminus V_1$.

This is done by first constructing a multigraph G' and then by analyzing a single agent movement in G'. The node set of G' is $\{h\} \cup (V \setminus V_1)$. For each $(u \to v) \in E_2 \setminus F$, let $(u \to v)$ be an edge of G', and for each $i \in \{1, \ldots, |F|\}$, let (h, v_i) and (v_i, h) be the edges of G'. In other words, we construct G' by taking G, leaving the edges in $E \setminus E_1$ untouched, and contracting (identifying) the nodes of H_1 into the single node h (see Fig. 2(b)). (The loops at h formed by the edges in E_1 are discarded.) For each $i \in \{1, \ldots .|F|\}$, the ports of $(h \to v_i)$ and $(v_i \to h)$ equal the ports of (u_i, v_i) and (v_i, u_i), respectively.

We set the remaining ports in G' and pointer initialization so that a single agent that starts at h explores G' in the following way:

(a) The edges in F are traversed according to the order

$$\big((h \to v_1), (v_1 \to h), (h \to v_2), (v_2 \to h), \ldots, (h \to v_{|F|}), (v_{|F|} \to h)\big).$$

Later on, we use the port labeling of G' to assign port labels to the edges in $E_2 \setminus F$ in G, and the above allows us to maintain the return property in G.

(b) The agent requires $D/2$ traversals through at least one edge in F (and $D/2 - 1$ through every other edge from F). This follows from the fact that, due to (iii), there exists a node in G' being at distance at least $D/2$ from h.

The above process assigns port labels to the edges in E_2 and sets initial values of all pointers in G', which completes the construction of G and the initial setup of the rotor-router.

Now we analyze the delayed deployment performed by the k agents in G. We divide the exploration of G into phases. The i-th phase starts in the step in which each edge in S_1 is traversed for the i-th time, and ends in the step preceding the beginning of the $(i+1)$-th stage. Note that each stage contains at least Δ steps in which all agent move simultaneously. By (a), the property of return holds in G, and therefore each edge in F is traversed exactly once in each of the phases except the 1st phase. (In the 1st phase, agents only traverse edges from E_1.) Thus, by (b), at least $D/2 - 1 + 1$ full phases are required in the delayed deployment to explore G (not counting the very last, partial phase in which the exploration of last vertex happens, but counting the initial phase in which no edges from F are traversed). This means that we need τ steps in which all agents move simultaneously to fully explore the graph G, where:

$$\tau \geq \Delta \cdot D/2 = \left\lfloor \tfrac{2|E_1|}{k} \right\rfloor \cdot D/2 \geq \lfloor \tfrac{m}{k} \rfloor \cdot D/2 \geq \tfrac{1}{4} mD/k$$

We can now apply Lemma 2.1 for the considered deployment, obtaining that the cover time of G is $t_C \geq \tau \geq \frac{1}{4} mD/k$. □

The bound in Theorem 4.1 is asymptotically tight, e.g., for the class of stars.

Proposition 4.2. *Let G be a star on n nodes. A team of $k \leq n$ agents covers G in time $t_C \leq 2\lceil n/k \rceil$, for any initialization of the rotor-router and any initial positions of agents.* □

Appendix A. Proof of Lemma 2.1

We introduce the following auxiliary notation. For a (possibly delayed) strategy $\mathcal{D}$, a node v and $t \geq 0$, $e_{\mathcal{D}}^{(t)}(v) = \sum_{u \in \Gamma(v)} d_{\mathcal{D}}^{(t)}(v \to u)$ denotes the total number of traversals of arcs outgoing from v until the end of step t for execution $\mathcal{D}$, and by $n_{\mathcal{D}}^{(t)}(v) = \sum_{t' \in [0,t]} |\{i : r_i^{(t)} = v\}|$ we denote the total number of visits of an agent at node v in time steps up to t inclusive; each visit is the result of edge traversal or initial placement. We start by making the following observation about fair strategies.

Lemma A.1. *For any (possibly delayed) fair strategy $\mathcal{D}$, any $t = 0, 1, \ldots$ and any $u \in V$ we have:*

$$e_{\mathcal{D}}^{(t+1)}(u) = n_{\mathcal{D}}^{(t)}(u) - \mathcal{D}(u, t+1).$$

Proof. Observe that for an arbitrary agent, the difference between the number of times the agent enters node u in rounds $[0,t]$ (either by traversing an arc entering u or due to the initial placement at u) and the number of times the agent leaves u in rounds $[0,t+1]$ by outgoing arcs is equal to either 1 or 0, depending on whether the agent is delayed at u in round $t+1$ or not. Summing over all agents, we obtain the claim. □

Lemma A.2. *Let $\mathcal{U}$ be a fair strategy, and let $\mathcal{D}_1, \mathcal{D}_2 \in delay(\mathcal{U})$ be two delayed deployments such that $\mathcal{D}_1(v,t) \geq \mathcal{D}_2(v,t)$ for all $v \in V$ and all rounds t. Then, for all $v \in V$ and all rounds t, we have $n^{(t)}_{\mathcal{D}_1}(v) \leq n^{(t)}_{\mathcal{D}_2}(v)$.*

Proof. The proof proceeds by induction on time t. First, let $t = 0$. By definition,

$$n^{(0)}_{\mathcal{D}_2}(v) - n^{(0)}_{\mathcal{D}_1}(v) = \sum_{w \in \Gamma(v)} d^{(0)}_{\mathcal{D}_2}(w \to v) - \sum_{w \in \Gamma(v)} d^{(0)}_{\mathcal{D}_1}(w \to v), \quad \text{for all } v \in V. \tag{16}$$

Since the same number of agents is initially placed in $\mathcal{D}_1$ and $\mathcal{D}_2$ at each node w, $\mathcal{D}_1$ and $\mathcal{D}_2$ have the same sequence of exits (because $\mathcal{D}_1$ and $\mathcal{D}_2$ correspond to the same undelayed fair strategy) and $\mathcal{D}_1(w,0) \geq \mathcal{D}_2(w,0)$, we obtain that $d^{(0)}_{\mathcal{D}_2}(w \to v) \geq d^{(0)}_{\mathcal{D}_1}(w \to v)$ for each $v \in V$ and $w \in \Gamma(v)$. Thus, by (16), the claim holds for $t = 0$.

Suppose that for some $t \geq 1$, $n^{(t-1)}_{\mathcal{D}_1}(v) \leq n^{(t-1)}_{\mathcal{D}_2}(v)$ holds for all $v \in V$. Then, we have from Lemma A.1:

$$e^{(t)}_{\mathcal{D}_1}(v) + \mathcal{D}_1(v,t) \leq e^{(t)}_{\mathcal{D}_2}(v) + \mathcal{D}_2(v,t)$$

and since, by assumption, $\mathcal{D}_1(v,t) \geq \mathcal{D}_2(v,t)$, we obtain

$$e^{(t)}_{\mathcal{D}_1}(v) \leq e^{(t)}_{\mathcal{D}_2}(v), \quad \text{for all } v \in V. \tag{17}$$

Now, since $\mathcal{D}_1$ and $\mathcal{D}_2$ have the same sequence of exits, (17) implies that

$$d^{(t)}_{\mathcal{D}_1}(v \to w) \leq d^{(t)}_{\mathcal{D}_2}(v \to w), \quad \text{for all arcs } v \to w \in \vec{E}.$$

Now, fix an arbitrary node u and observe that the number of visits to node u within the interval $[0,t]$ is equal to the sum of the number of agents placed at u in round 0 (which equals $i_j = n^{(0)}_{\mathcal{D}_j}(u) - \sum_{w \in \Gamma(u)} d^{(0)}_{\mathcal{D}_j}(w \to u)$ for deployment $\mathcal{D}_j$, $j \in \{1,2\}$), and the number of times an agent exited one of its neighbors $v \in \Gamma(u)$ along an arc $v \to u$ in rounds $[0,t]$. Therefore,

$$n^{(t)}_{\mathcal{D}_1}(u) = i_1 + \sum_{w \in \Gamma(u)} d^{(t)}_{\mathcal{D}_1}(w \to u) \leq i_2 + \sum_{w \in \Gamma(u)} d^{(t)}_{\mathcal{D}_2}(w \to u) = n^{(t)}_{\mathcal{D}_2}(u),$$

because $i_1 = i_2$ for deployments $\mathcal{D}_1$ and $\mathcal{D}_2$ of the same fair strategy. □

Lemma A.3. *Let $\mathcal{U}$ be a fair strategy and let $\mathcal{D} \in delay(\mathcal{U})$. Let T be any fixed time round, and let τ be the number of rounds in the interval $[0,T]$ such that all the agents are active in $\mathcal{D}$, i.e., $\tau = |\{t \in [0,T] : \forall_{v \in V}\ \mathcal{D}(v,t) = 0\}|$. Then, for all nodes $v \in V$, we have: $n^{(\tau)}_{\mathcal{U}}(v) \leq n^{(T)}_{\mathcal{D}}(v) \leq n^{(T)}_{\mathcal{U}}(v)$.*

Proof. Let $v \in V$ be selected arbitrarily. The right inequality follows directly from Lemma A.2. Consider a function $f : [0,\tau] \to [0,T]$, with $f(i)$ being the i-th time round in which all agents are active in delayed deployment $\mathcal{D}$.

To prove the left inequality, we argue by induction on $i = 0, 1, \ldots, \tau$ that

$$n^{(f(i))}_{\mathcal{D}}(v) \geq n^{(i)}_{\mathcal{U}}(v). \tag{18}$$

For $i = 0$, since $\mathcal{D}$ and $\mathcal{U}$ have the same initialization of agents and $\mathcal{D}$ is delayed with respect to $\mathcal{U}$, it holds that $n^{(f(0))}_{\mathcal{D}}(v) \leq n^{(0)}_{\mathcal{U}}(v)$.

Assume now that (18) holds for some for some $i \geq 0$. Equation (18) provides a relation between the numbers of exits from nodes in both processes till the end of round $f(i+1)$. More precisely,

$$e^{(f(i+1))}_{\mathcal{D}}(v) = n^{(f(i+1)-1)}_{\mathcal{D}}(v) \geq n^{(f(i))}_{\mathcal{D}}(v) \geq n^{(i)}_{\mathcal{U}}(v) = e^{(i+1)}_{\mathcal{U}}(v),$$

where the first equality is due to the fact that no agents are stopped at v in round $f(i+1)$ in $\mathcal{D}$.

Therefore, since number of exists from each vertex in $\mathcal{D}$ in not smaller than in $\mathcal{U}$ and both deployments have the same sequence of exits, we obtain a relation between a number of traversals of directed arcs. For all arcs $u \to v \in \vec{E}$,

$$d^{f(i+1)}_{\mathcal{D}}(u \to v) \geq d^{(i+1)}_{\mathcal{U}}(u \to v).$$

Finally, we obtain a correspondence between the number of visits at v in $\mathcal{D}$ and $\mathcal{U}$ by taking into account that the number of visits is correlated with the number of traversals of incoming arcs:

$$n_{\mathcal{D}}^{(f(i+1))}(v) = i_{\mathcal{D}} + \sum_{w \in \Gamma(v)} d_{\mathcal{D}}^{(f(i+1))}(w \to v) \geq i_{\mathcal{U}} + \sum_{w \in \Gamma(v)} d_{\mathcal{U}}^{(i+1)}(w \to v) = n_{\mathcal{U}}^{(i+1)}(v),$$

where $i_{\mathcal{D}}$ and $i_{\mathcal{U}}$ are the numbers of agents initially placed at v in $\mathcal{D}$ and $\mathcal{U}$, respectively. (Note that $i_{\mathcal{D}} = i_{\mathcal{U}}$.) This completes the proof of (18).

By taking $i = \tau$ in (18), we obtain the sought inequality $n_{\mathcal{D}}^{(T)}(v) \geq n_{\mathcal{D}}^{(f(\tau))}(v) \geq n_{\mathcal{U}}^{(\tau)}(v)$. □

Lemma A.3 immediately implies the claim of Lemma 2.1.

References

[1] H. Akbari, P. Berenbrink, Parallel rotor walks on finite graphs and applications in discrete load balancing, in: SPAA, 2013, pp. 186–195.
[2] N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, M.R. Tuttle, Many random walks are faster than one, Comb. Probab. Comput. 20 (4) (2011) 481–502.
[3] E. Bampas, L. Gasieniec, N. Hanusse, D. Ilcinkas, R. Klasing, A. Kosowski, Euler tour lock-in problem in the rotor-router model, in: DISC, 2009, pp. 423–435.
[4] E. Bampas, L. Gasieniec, R. Klasing, A. Kosowski, T. Radzik, Robustness of the rotor-router mechanism, in: OPODIS, in: LNCS, vol. 5923, 2009, pp. 345–358.
[5] S.N. Bhatt, S. Even, D.S. Greenberg, R. Tayar, Traversing directed eulerian mazes, J. Graph Algorithms Appl. 6 (2) (2002) 157–173.
[6] A.Z. Broder, A.R. Karlin, P. Raghavan, E. Upfal, Trading space for time in undirected s-t connectivity, in: STOC, 1989, pp. 543–549.
[7] C. Cooper, D. Ilcinkas, R. Klasing, A. Kosowski, Derandomizing random walks in undirected graphs using locally fair exploration strategies, Distrib. Comput. 24 (2) (2011) 91–99.
[8] J.N. Cooper, J. Spencer, Simulating a random walk with constant error, Comb. Probab. Comput. 15 (6) (2006) 815–822.
[9] D. Dereniowski, A. Kosowski, D. Pajak, P. Uznanski, Bounds on the cover time of parallel rotor walks, in: LIPIcs, in: STACS, vol. 25, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014, pp. 263–275.
[10] B. Doerr, T. Friedrich, Deterministic random walks on the two-dimensional grid, Comb. Probab. Comput. 18 (1–2) (2009) 123–144.
[11] K. Efremenko, O. Reingold, How well do random walks parallelize?, in: APPROX-RANDOM, 2009, pp. 476–489.
[12] R. Elsässer, T. Sauerwald, Tight bounds for the cover time of multiple random walks, Theor. Comput. Sci. 412 (24) (2011) 2623–2641.
[13] U. Feige, A spectrum of time–space trade-offs for undirected s-t connectivity, J. Comput. Syst. Sci. 54 (2) (1997) 305–316.
[14] T. Friedrich, T. Sauerwald, The cover time of deterministic random walks, in: COCOON, in: LNCS, vol. 6196, 2010, pp. 130–139.
[15] A.E. Holroyd, J. Propp, Rotor walks and Markov chains, in: Algorithmic Probability and Combinatorics, American Mathematical Society, 2010, pp. 904–4507.
[16] R. Klasing, A. Kosowski, D. Pajak, T. Sauerwald, The multi-agent rotor-router on the ring: a deterministic alternative to parallel random walks, in: PODC, 2013, pp. 365–374.
[17] S. Koenig, R.G. Simmons, Easy and hard testbeds for real-time search algorithms, AAAI 96 and IAAI 96, vol. 1, AAAI Press/The MIT Press, 1996, pp. 279–285.
[18] A. Kosowski, D. Pajak, Does adding more agents make a difference? A case study of cover time for the rotor-router, in: ICALP, in: Lecture Notes in Computer Science, vol. 8573, Springer, 2014, pp. 544–555.
[19] L. Levine, Y. Peres, Spherical asymptotics for the rotor-router model in $\mathbb{Z}^d$, Indiana Univ. Math. J. 57 (2008) 431–450.
[20] L. Levine, Y. Peres, The rotor-router shape is spherical, Math. Intell. 27 (3) (2005) 9–11.
[21] L. Levine, Y. Peres, Strong spherical asymptotics for rotor-router aggregation and the divisible sandpile, 2007.
[22] V. Priezzhev, D. Dhar, A. Dhar, S. Krishnamurthy, Eulerian walkers as a model of self-organized criticality, Phys. Rev. Lett. 77 (25) (Dec 1996) 5079–5082.
[23] T. Sauerwald, Expansion and the cover time of parallel random walks, in: PODC, ACM, 2010, pp. 315–324.
[24] T. Shiraga, Y. Yamauchi, S. Kijima, M. Yamashita, Deterministic random walks for rapidly mixing chains, arXiv:1311.3749 [CoRR], 2013.
[25] I.A. Wagner, M. Lindenbaum, A.M. Bruckstein, Distributed covering by ant-robots using evaporating traces, IEEE Trans. Robot. Autom. 15 (1999) 918–933.
[26] V. Yanovski, I.A. Wagner, A.M. Bruckstein, A distributed ant algorithm for efficiently patrolling a network, Algorithmica 37 (3) (2003) 165–186.