Bounds on the Cover Time of Parallel Rotor Walks*

Dariusz Dereniowski[1], Adrian Kosowski[2,3], Dominik Pająk[2], and Przemysław Uznański[2,4]

1 **Department of Algorithms and System Modeling, Gdańsk University of Technology, Gdańsk, Poland**
2 **CEPAGE Project, Inria Bordeaux Sud-Ouest – LaBRI, Talence, France**
3 **GANG Project, Inria Paris Rocquencourt – LIAFA, Paris, France**
4 **CNRS and Aix-Marseille Université – LIF, Marseille, France**

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 maintains a cyclic ordering of its outgoing arcs, and successively propagates walkers which visit it along its outgoing arcs in round-robin fashion, according to the 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 starting locations of the walks. In the case of $k = 1$, Yanovski et al. (2003) and Bampas et al. (2009) showed that a single walk achieves a cover time of exactly $\Theta(mD)$ for any n-node graph with m edges and diameter D, and that the walker eventually stabilizes to a traversal of an Eulerian circuit on the set of all directed edges of the graph. For $k > 1$ parallel walks, no similar structural behaviour can be observed.

In this work we provide tight bounds on the cover time of k parallel rotor walks in a graph. We show that this cover time is at most $\Theta(mD/\log k)$ and at least $\Theta(mD/k)$ for any graph, which corresponds to a speedup of between $\Theta(\log k)$ and $\Theta(k)$ with respect to the cover time of a single walk. Both of these extremal values of speedup are achieved for some graph classes. Our results hold for up to a polynomially large number of walks, $k = O(poly(n))$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Distributed graph exploration, Rotor-Router, Collaborative robots, Parallel random walks, Derandomization

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.263

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

* 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 founded by the Polish Ministry of Science and Higher Education. The full text of the paper is available online at: `http://hal.inria.fr/hal-00865065`.

in little memory rely on the deployment of multiple random walks [6, 11]. 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 [9], and Elsässer and Sauerwald [10] 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 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 speed-up 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.* [14], 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.

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 $\Theta(mD)$ [16, 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.

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.* [16] 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.* [13] 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.

1.1 Related work

The rotor-router model. Studies of the rotor-router started with works of Wagner *et al.* [15] 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 $\vec{G}$ (see Section 3 for a definition). Subsequently, Yanovski *et al.* [16] 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 [12]. Robustness properties of the rotor-router were further studied in [4], who 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* [15, 16], 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 was performed by Yanovski *et al.* [16] and Klasing *et al.* [13], 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. The single agent is replaced with a number of agents, referred to as *tokens*. Cooper and Spencer [7] study d-dimensional grid graphs and show 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 [8] analyze 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.

Parallel random walks. Alon *et al.* [2] introduced the notion of the speed-up 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 speed-up 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, 10], hypercubes [2, 10], expanders [2, 10], and different models of random graphs [2, 10]. For the cycle, the speed-up is equal to $\log k$ [2]. For general graphs, an upper bound $\min\{k \log n, k^2\}$ on the speed-up was obtained by Efremenko *et al.* [9]. Independently, Elsässer *et al.* [10] showed the $k \log n$ upper bound. Another measure studied by Efremenko *et al.* [9] concerns the speedup with respect to a different exploration parameter — the maximing 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 show 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.

We start by providing a formal definition of the rotor-router model and recalling its basic properties in Section 2. In Section 3, we first prove that the cover time t_C satisfies $t_C \in O(mD/\log k)$, when $k < 2^{16D}$. We then extend this result to the case of $k \in O(poly(n))$, i.e., $k < n^c$ for some absolute constant c. 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.* [16], and apply them within a global

Table 1 Values of speed-up for k-agent exploration with the rotor-router and parallel random walks. All results hold at least for $k \leq n$, except for those cited from [13] which hold for $k \leq n^{1/11}$.

Graph class	Speedup of Rotor-Router		Speedup of Random Walk			
	for cover time		*for cover time*		*for max hitting time*	
General case:	$\Omega(\log k)$, $O(k)$	**(Thm. 8, 9)**	$O(k^2)$, $O(k \log n)$	[9, 10]	$O(k)$	[10]
Cycle:	$\Theta(\log k)$	[13]	$\Theta(\log k)$	[2]	$\Theta(\log k)$	[2]
Star:	$\Theta(k)$	**(Prop. 10)**	$\Theta(k)$	[2]	$\Theta(k)$	[2]

amortization argument over all time steps and all edges in the graph. The extension to the case of $k \in O(poly(n))$ 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.* [13], which we briefly recall in Section 2. We remark that by [13], a cover time of $\Theta(mD/\log k)$ is achieved when G is a cycle with all agents starting from one node, when $k < n^{1/11}$.

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 speed-up 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 $\vec{E} = \{(v, u) : \{v, u\} \in E\}$. We will denote arc (v, u) by $v \to u$.

Model definition. We consider the rotor-router model (on graph G) with $k \geq 1$ indistinguishable agents, which run in steps, synchronized by a global clock. In each step, 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 arc next 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 rounds, 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 $(v, next(\pi_v))$.

Notation. 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 $a, b \in \mathbb{N}$.

We will denote by $a^{(t)}(e)$ the number of agents traversing directed arc $e \in \vec{E}$ during step $t+1$. We recall that multiple agents traversing one arc $e \in \vec{E}$ in the same time step t 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 $u \to v \in \vec{E}$, $0 \le d^{(t)}(u \to v) - d^{(t)}(u) \le 1$ [16].

We also denote $V_i^{(t)} = \{v \in V : d^{(t)}(v) \le i\}$ and $E_i^{(t)} = \{e \in \vec{E} : d^{(t)}(e) \le i\}$. Given a graph $G = (V, E)$ and a subset $X \subseteq V$, $G[X]$ denotes the subgraph of G *induced by* X, $G[X] = (X, \{\{u, v\} \in E \mid u, v \in X\})$.

Delayed deployment technique. In some of the proofs, we will make use of modified executions of the k-agent rotor-router system called *delayed deployments* [13], in which some agents may be stopped at a node, skipping their move for some number of rounds. Formally, a delayed deployment D of k agents is defined as a function $D : V \times \mathbb{N} \to \mathbb{N}$, where $D(v, t) \ge 0$ represents the number of agents which are stopped in node v in step t of the execution of the system. Delayed deployments may be conveniently viewed as algorithmic procedures for delaying agents, and are introduced for purposes of analysis, only. The following lemma relates the cover time of the rotor-router system to that of its delayed deployment.

▶ **Lemma 1.** *[13] Let R be a k-rotor router system with an arbitrarily chosen initialization, and let D be any delayed deployment of R. Suppose that deployment D covers all the nodes of the graph after T rounds, and in at least τ of these rounds, all k agents were active in D. Then, the cover time t_C of the rotor-router system R can be bounded by: $\tau \le t_C \le T$.*

3 Upper bound on cover time

In this section, we will show that a k-agent parallel rotor-router system explores a 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_C. We look at the number of visits to consecutive arcs on this path. It turns out that the rotor-router 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 e_l. In this way, the larger the discrepancy between the number of visits to adjacent arcs, the more activity will the rotor-router 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 $(\log k)$-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.* [16, 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 2.** *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 (set T) has completed at most d (more than d) full cycles of if its rotor, $S = V_d^{(t)}$ and $T = V \setminus S$. Suppose that for some nodes $v \in S$, $u \in T$, and some $\delta \in \mathbb{N}$, there exists an arc $u \to v$, such that $d^{(t)}(u \to v) \geq d + \delta$. Then, the set of arcs having their tail at a node of S will be traversed by at least $\delta - 1$ agents in total in step $t + 1$.*

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.** *Let $G = (V, E)$ be any undirected graph 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 $x \geq d^{(t)}(e_1) + 1$, the number of agents that traverse arcs from set $E_x^{(t)}$ in time step $t + 1$ satisfies:*

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

The property of the rotor-router captured by the above lemma is, in fact, sufficient to prove the main results of the 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 4.** *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 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 rotor-router 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:

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

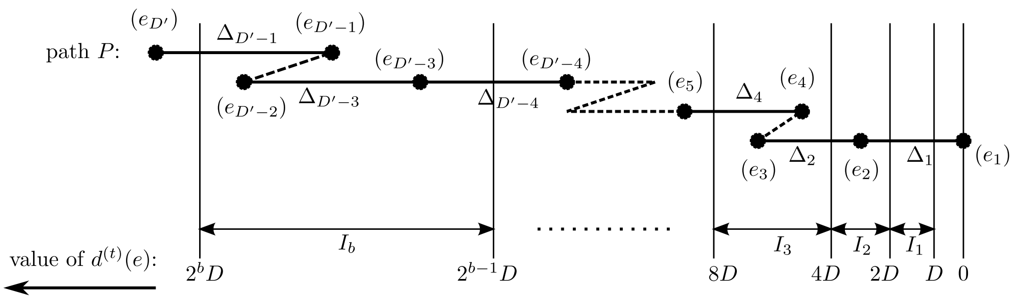

Figure 1 An illustration of sets I_i and Δ_l in the proof of Theorem 4.

We now analyze which buckets successive arcs of the path $\mathcal{P}$ fall into. 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 $[0, 2^bD)$, since for any $x \in [0, 2^bD)$ there exists $l^* \in [1, D')$ such that $x \in \Delta_{l^*}$ because $d^{(t)}(e_1) = 0$ and $d^{(t)}(e_{D'}) \geq 2^bD$ (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 $[0, 2^bD)$ 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 intervals 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 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 $x \in [0, 2^bD)$, and let l^* be such that $x \in \Delta_{l^*}$. We have $d^{(t)}(e_{l^*}) \leq x < 2^bD$, hence $l^* \in I_{i^*}$, for some $i^* \in [1, b]$, and $x \in \mathcal{X}_{i^*}$. It follows that:

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

For $i \in \mathbb{N}$, denote by $a_i^{(t)}$ the number of agents that traverse arcs from set $E_{2^iD}^{(t)}$ in step $t+1$, $a_i^{(t)} \equiv \sum_{e \in E_{2^iD}^{(t)}} a^{(t)}(e)$, and let $a_{-1}^{(t)} = 0$. (We remark that $E_{2^iD}^{(t)} \supseteq I_1 \cup \ldots \cup I_i$.) First, note that for all $i \in [1, b]$ and for $l \in I_i$, we have $d^{(t)}(e_l) < 2^iD$. So, by Lemma 3:

$$a_i^{(t)} \geq d^{(t)}(e_{l+1}) - d^{(t)}(e_l) - 1 = |\Delta_l| - 1 \quad \Longrightarrow \quad |\Delta_l| \leq a_i^{(t)} + 1. \tag{2}$$

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

$$\max \mathcal{X}_i = \max_{l \in I_i} (\max \Delta_l) \leq \max_{l \in I_i} \left(d^{(t)}(e_l) + |\Delta_l| - 1 \right) < 2^iD + a_i^{(t)}, \tag{3}$$

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

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

$$2^{i-1}D - a_{i-1}^{(t)} \leq |\mathcal{X}_i| \leq |I_i|(a_i^{(t)} + 1). \tag{4}$$

The right inequality in (4) is proved as follows: $|\mathcal{X}_i| \le \sum_{l \in I_i} |\Delta_l| \le |I_i|(a_i^{(t)} + 1)$, where the latter inequality is a consequence of (2).

We now prove the left inequality in (4). If $a_{i-1}^{(t)} \ge 2^{i-1}D$, then the bound is trivial. In the case when $a_{i-1}^{(t)} < 2^{i-1}D$, we will first prove that:

$$[2^{i-1}D + a_{i-1}, 2^i D) \subseteq \mathcal{X}_i. \tag{5}$$

To this end, take any $x \in [2^{i-1}D + a_{i-1}, 2^i D)$ and observe that by (1), 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 (3), $\max \mathcal{X}_j < 2^j D + a_j^{(t)} \le 2^{i-1}D + a_{i-1}^{(t)} \le x$.
2. For any $j > i$, $x \notin \mathcal{X}_j$, because: $\min \mathcal{X}_j = \min_{l \in I_j, \Delta_l \ne \emptyset} \min \Delta_l = \min_{l \in I_j, \Delta_l \ne \emptyset} d^{(t)}(e_l) \ge 2^{j-1}D \ge 2^i D > x$.

Thus, $x \in \mathcal{X}_i$, and (5) follows. Equation (5) implies that $|\mathcal{X}_i| \ge 2^{i-1}D - a_{i-1}^{(t)}$, which completes the proof of (4). Next, by (4), $|I_i| \ge \frac{2^{i-1}D - a_{i-1}^{(t)}}{a_i^{(t)}+1}$ 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:

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

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

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

where in the last inequality we took into account that $k \le 2^{16D}$ and $b \le (\log k)/2$ by assumption, hence $D \ge (\log k)/16 \ge b/8$, and that $b = \lfloor (\log k)/2 \rfloor \ge 80$. 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 $|S^{(t)}| \ge b/2$, such that for all $j \in S^{(t)}$ we have:

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

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

$$a_j^{(t)} \ge \tfrac{b}{18.4} \cdot 2^{j-1} - 1 > \tfrac{b}{25} \cdot 2^{j-1}, \tag{6}$$

where we again took into account that $b \ge 80$. Fix $t_1 = \lceil 100mD/b \rceil$. We now prove that

$$t_C \le t_0 + 2t_1 + 4m. \tag{7}$$

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 $|S^{(t)}| \ge b/2$ 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 visits to arc e of the graph for the first time exceeds $2^{j^*}D$: $d^{(t_e)}(e) \le 2^{j^*}D < d^{(t_e+1)}(e)$. 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 $t \le t_e$, and therefore we may write:

$$\sum_{t\in[0,t_C)\setminus\hat{T}} a^{(t)}_{j^*} = \sum_{t\in[0,t_C)\setminus\hat{T}} \sum_{e\in E^{(t)}_{2^{j^*}D}} a^{(t)}(e) \le \sum_{e\in\vec{E}} \sum_{t=0}^{t_e-1} a^{(t)}(e) = \sum_{e\in\vec{E}} d^{(t_e)}(e) \le 2m\cdot 2^{j^*}D. \quad (8)$$

Now, recall that there are at least $t_1 + 2m$ good time steps $t \in [t_0, t_C)$ for which index j^* satisfies (6), and that $|\hat{T}| \le 2m$. It follows that:

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

a contradiction with (8). Thus, we have proved (7). By (7), we obtain

$$\begin{aligned} t_C &\le t_0 + 2t_1 + 4m = \left\lceil \frac{2^{b+1}mD}{k} \right\rceil + 2\left\lceil \frac{100mD}{b} \right\rceil + 4m \le \\ &\le \frac{mD}{\log k}\left(\frac{2^{b+1}\log k}{k} + \frac{200\log k}{b} + \frac{4\log k}{D} + \frac{3\log k}{mD} \right) \end{aligned} \quad (9)$$

Taking into account that $b = \lfloor (\log k)/2 \rfloor$, $k \le 2^{16D}$, 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 $k \le 2^{160}$. Yanovski *et al.* [16] 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: $t_C \le 2mD < 500\frac{mD}{\log k}$. ◀

We now consider the case when $k \ge 2^{16D}$. Here, we first make the additional assumption that each agent starts from a distinct node. We show that 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 4, 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 $D' \le D + 2$ is small compared to $\log k$, which can be used to infer the existence of the sought arc pairs.

▶ **Lemma 5.** *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. If $k \ge 2^{16D}$, then a team of k agents performing parallel rotor-router movement, with each agent starting from a distinct node of the graph, explores G in time $16mD/\log k$.*

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 6.** *Let R and R' be two starting configurations of the k-agent rotor-router system with cover times t_C and t'_C, respectively. Suppose that there exists a delayed deployment D of R whose execution transforms the starting configuration of R into the starting configuration of R' in $\hat{t}$ time steps. Then, $t_C \le \hat{t} + t'_C$.*

The next lemma provides an upper bound on the time of transforming a rotor-router configuration with at most n agents into one in which agents occupy distinct starting nodes.

▶ **Lemma 7.** *For any initialization R of the rotor-router system with k agents, $k \leq n$, there exists a delayed deployment D of R which terminates in a configuration in which all agents occupy distinct positions after $\hat{t} \leq k^4$ steps.*

When $1 < k \leq \lceil n^{1/5} \rceil$, we can bound the time $\hat{t}$ in the above lemma as: $\hat{t} \leq k^4 \leq 32n/k \leq 64m/k \leq 128\frac{mD}{\log k}$.

Combining the above result with Lemmas 5 and 6, we obtain that for any rotor router initialization with k agents, $k \leq \lceil n^{1/5} \rceil$ and $k \geq 2^{16D}$, exploration is completed within time $t_C = \hat{t} + t'_C \leq 128\frac{mD}{\log k} + 16\frac{mD}{\log k} = 144\frac{mD}{\log k}$. On the other hand, when $k < 2^{16D}$, by Theorem 4, 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.* [16], stating that the worst-case initialization of a rotor-router 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 8.** *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 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* $2500c \cdot mD/\log k$. ◀

Theorems 4 and 8 imply that the cover time of the rotor-router is $O(mD/\log k)$ for all graphs, whenever $k \in 2^{O(D)}$ or $k \in O(poly(n))$.

4 Lower bound on cover time

▶ **Theorem 9.** *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.

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:

$$\forall_{1 \leq i < k} \quad j_{i+1} - j_i \in \{\Delta, \Delta + 1\} \quad \text{and} \quad j_1 - j_k + 2|E_1| \in \{\Delta, \Delta + 1\}\,.$$

This is possible because, due to 1, $2|E_1| \geq k$. We partition E_1 into Δ sets $S_1, \ldots, S_\Delta$ of size k:

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

and one set for all remaining edges: $R = E_1 \setminus \bigcup_{t=1}^{\Delta} S_t$.

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 2, 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 pointers for the nodes in $V \setminus V(\widetilde{G})$. 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. (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 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 3, 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 3a, 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 3b, 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:

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

We can now apply Lemma 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 9 is asymptotically tight, e.g., for the class of stars.

▶ **Proposition 10.** 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. ◀

References

1 Hoda Akbari and Petra Berenbrink. Parallel rotor walks on finite graphs and applications in discrete load balancing. In *SPAA*, pages 186–195, 2013.

2 Noga Alon, Chen Avin, Michal Koucký, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. *Combinatorics, Probability & Computing*, 20(4):481–502, 2011.

3 Evangelos Bampas, Leszek Gasieniec, Nicolas Hanusse, David Ilcinkas, Ralf Klasing, and Adrian Kosowski. Euler tour lock-in problem in the rotor-router model. In *DISC*, pages 423–435, 2009.

4 Evangelos Bampas, Leszek Gasieniec, Ralf Klasing, Adrian Kosowski, and Tomasz Radzik. Robustness of the rotor-router mechanism. In *OPODIS*, volume 5923 of *LNCS*, pages 345–358, 2009.

5 S. N. Bhatt, S. Even, D. S. Greenberg, and R. Tayar. Traversing directed eulerian mazes. *J. Graph Algorithms Appl.*, 6(2):157–173, 2002.

6 Andrei Z. Broder, Prabhakar Raghavan, Robert W. Taylor, Anna R. Karlin, Anna R. Karlin, Eli Upfal, and Eli Upfal. Trading space for time in undirected s-t connectivity. In *In Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 543–549, 1991.

7 J. N. Cooper and J. Spencer. Simulating a random walk with constant error. *Combinatorics, Probability & Computing*, 15(6):815–822, 2006.

8 B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Combinatorics, Probability & Computing*, 18(1-2):123–144, 2009.

9 Klim Efremenko and Omer Reingold. How well do random walks parallelize? In *APPROX-RANDOM*, pages 476–489, 2009.

10 Robert Elsässer and Thomas Sauerwald. Tight bounds for the cover time of multiple random walks. *Theor. Comput. Sci.*, 412(24):2623–2641, 2011.

11 Uriel Feige. A Spectrum of Time–Space Trade-offs for Undirected s-t Connectivity. *Journal of Computer and System Sciences*, 54(2):305 – 316, 1997.

12 Tobias Friedrich and Thomas Sauerwald. The cover time of deterministic random walks. In *COCOON*, volume 6196 of *LNCS*, pages 130–139, 2010.

13 Ralf Klasing, Adrian Kosowski, Dominik Pajak, and Thomas Sauerwald. The multi-agent rotor-router on the ring: a deterministic alternative to parallel random walks. In *PODC*, pages 365–374, 2013.

14 V.B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy. Eulerian walkers as a model of self-organized criticality. *Phys. Rev. Lett.*, 77(25):5079–5082, Dec 1996.

15 I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Trans. Robotics and Automation*, 15:918–933, 1999.

16 V. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.