Distrib. Comput. (2017) 30:127–148
DOI 10.1007/s00446-016-0282-y

# The multi-agent rotor-router on the ring: a deterministic alternative to parallel random walks

**Ralf Klasing**[1] · **Adrian Kosowski**[2] · **Dominik Pająk**[3] · **Thomas Sauerwald**[4]

Received: 29 January 2014 / Accepted: 30 August 2016 / Published online: 17 September 2016


**Abstract** The *rotor-router mechanism* was introduced as a deterministic alternative to the random walk in undirected graphs. In this model, an agent is initially placed at one of the nodes of the graph. Each node maintains a cyclic ordering of its outgoing arcs, and during successive visits of the agent, propagates it along arcs chosen according to this ordering in round-robin fashion. The behavior of the rotor-router is fully deterministic but its performance characteristics (cover time, return time) closely resemble the expected values of the corresponding parameters of the random walk. In this work we consider the setting in which multiple, indistinguishable agents are deployed in parallel in the nodes of the graph, and move around the graph in synchronous rounds, interacting with a single rotor-router system. We propose new techniques which allow us to perform a theoretical analysis of the multi-agent rotor-router model, and to compare it to the scenario of parallel independent random walks in a graph. Our main results concern the $n$-node ring, and suggest a strong similarity between the performance characteristics of this deterministic model and random walks. We show that on the ring the rotor-router with $k$ agents admits a cover time of between $\Theta(n^2/k^2)$ in the best case and $\Theta(n^2/\log k)$ in the worst case, depending on the initial locations of the agents, and that both these bounds are tight. The corresponding expected value of the cover time for $k$ random walks, depending on the initial locations of the walkers, is proven to belong to a similar range, namely between $\Theta(n^2/(k^2/\log^2 k))$ and $\Theta(n^2/\log k)$. Finally, we study the limit behavior of the rotor-router system. We show that, once the rotor-router system has stabilized, all the nodes of the ring are always visited by some agent every $\Theta(n/k)$ steps, regardless of how the system was initialized. This asymptotic bound corresponds to the expected time between successive visits to a node in the case of $k$ random walks. All our results hold up to a polynomially large number of agents ($1 \leq k < n^{1/11}$).

Research partially supported by the ANR Project DISPLEXITY (ANR-11-BS02-014). This study has been carried out in the frame of the Investments for the future Programme IdEx Bordeaux—CPU (ANR-10-IDEX-03-02). Research partially supported by the NCN under contract 2015/17/B/ST6/01897. Some of the results of this paper appeared in the extended abstract [20], published in the Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC 2013).

✉ Dominik Pająk
dominik.pajak@pwr.edu.pl

Ralf Klasing
ralf.klasing@labri.fr

Adrian Kosowski
adrian.kosowski@inria.fr

Thomas Sauerwald
thomas.sauerwald@cl.cam.ac.uk

[1] LaBRI, CNRS, Université de Bordeaux, 33400 Talence, France

[2] IRIF, CNRS, Inria, Université Paris Diderot, 75013 Paris, France

[3] Department of Computer Science, Faculty of Fundamental Problems of Technology, Wroclaw University of Science and Technology, 50-370 Wrocław, Poland

[4] Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, UK

## 1 Introduction

The study of deterministic exploration strategies in agent-based models of computation is largely inspired by considerations of random walk processes. For an undirected graph $G = (V, E)$, exploration with the random walk has many advantageous properties: the expected time for the agent to visit all nodes of the graph, known as the *cover time* $C(G)$,

can in general be bounded as, e.g., $C(G) \in O(D|E|\log|V|)$, where $D$ is the diameter of the graph. The random walk also has the property that in the limit it visits all of the edges of the graph with the same frequency, on average, traversing each once every $|E|$ rounds. The rotor-router model, introduced by Priezzhev et al. [23] and further popularized by James Propp, provides a mechanism for the environment to control the movement of the agent deterministically, whilst retaining similar properties of exploration as the random walk.

In the rotor-router model, 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 $v$.

The behavior of the rotor-router for a single agent is well understood. Yanovski et al. [27] showed that, regardless of the initialization of the system, the agent stabilizes to a traversal of a directed Eulerian cycle (containing all of the edges of the graph) within $2D|E|$ steps. A complementary lower bound was provided by Bampas et al. [6], who showed that for any graph there exists an initialization of the system for which covering all the nodes of the graph and entering the Eulerian cycle takes $\Theta(D|E|)$ steps. Despite seemingly similar general-case bounds on the cover time for the random walk and the rotor-router, there exist graphs for which these times differ. For example, for the two-dimensional square grid the rotor-router covers all nodes in $\Theta(|V|^{3/2})$ rounds in the worst case, while the cover time of the random walk is $\Theta(|V|\log^2|V|)$.

Our work deals with the problem of exploring a graph with the *multi-agent rotor-router*, i.e., a rotor-router system in which more than one agent are deployed in the same environment. Due to the interaction of the agents, which move the same set of pointers at nodes, this can be seen as an example of a deterministic interacting particle system. We compare our results with the so-called *parallel random walk*, achieved by deploying independent agents performing random walks in a graph independently and without any form of coordination. Recent work on the area of parallel random walks [4,14,15,25] contains a characterization of the improvement of the cover time due to the deployment of $k$ independent random walkers with respect to the case with a single walker. It is shown in these works that the achieved speed-up depends on different parameters, such as the mixing time [15] and edge expansion [25] of the graph. The speed-up may sometimes be as low as $\Theta(\log k)$ [4], and sometimes as high as exponential in terms of $k$ [14]. For many classes of graphs the speed-up is linear in terms of $k$ (especially when $k$ is small, $k \in O(\log n)$ [4]).

### 1.1 Our results and organization of the paper

In this work, we perform a comparative case study of two seemingly different scenarios: deterministic exploration with interacting particles in the rotor-router model vs. randomized exploration with non-interacting particles in the random walk, showing certain similarities between them.

We focus on two parameters of exploration. The first is the *cover time*, understood as the time before each node of the graph is covered by at least one agent. The second is the *return time*, i.e., the longest time during which some node remains unvisited in the limit, disregarding the time until the rotor-router enters its limit cycle. Note that the rotor-router, as a deterministic finite-state system, has to stabilize to a cyclic traversal of some set of configurations on the graph (where two configurations differ if some pointer or some agent is on different position). We present our results taking into account different initial locations of the set of agents.

In Sect. 2, we formally describe the model of the rotor-router system, and introduce the techniques used in the analysis of the multi-agent rotor-router system. The basic tool is applicable to general graphs and gives us an algorithmic perspective for analysis of the rotor-router through *delayed deployments* (Sect. 2.1), allowing the occasional stopping of some of the agents without affecting asymptotic cover time. For the specific case of the rotor-router on the ring (cycle), we describe states in the evolution of the system in which particular agents cover nearly disjoint, dynamically changing parts of the graph, known as *agent domains* (Sect. 2.2). We also introduce a *continuous time approximation* of the evolution of the system on the ring (Sect. 2.3), which allows us to postulate an asymptotic description of the behavior of the agents on the ring. Formal proofs of correctness are obtained through an analysis of the motion of agents within their domains in delayed deployments of the rotor-router.

Our main results for the case when the explored graph is a ring are presented in Sect. 3 (cf. Table 1 for an overview). We show that for a $k$-agent rotor-router system, the cover time is between $\Theta(n^2/k^2)$ and $\Theta(n^2/\log k)$, depending on the initial placement of the agents in the rotor-router. The first bound is achieved, in particular, for agents distributed uniformly on the ring, while the latter for agents initially located on the same node of the ring. The return time of the $k$-agent rotor-router on the ring is determined in Sect. 4 as $\Theta(n/k)$.

We remark that for a single agent, the rotor-router on the ring deterministically achieves a cover time of $\Theta(n^2)$, which matches that of the random walk. As the number of agents $k$ increases, the speed-up of the rotor-router with

**Table 1** The cover time of the multi-agent roter-router on the ring compared to multiple random walks (for $k < n^{1/11}$)

| Model | Cover time | | Return time |
|---|---|---|---|
| | For worst placement | For best placement | |
| $k$-Agent rotor-router | $\Theta(n^2/\log k)$ Theorems 1, 2 | $\Theta(n^2/k^2)$ Theorems 3, 4 | $\Theta(n/k)$ Theorem 6 |
| $k$ Random walks (expectations) | $\Theta(n^2/\log k)$ [4] | $\Theta\left(n^2 \Big/ \frac{k^2}{\log^2 k}\right)$ Theorem 5 | $\Theta(n/k)$ e.g. [2] |

respect to a single-agent system is seen from our results as between $\Theta(\log k)$ and $\Theta(k^2)$, depending on the initialization. These results are comparable with the corresponding speed-up of the random walk, which is between $\Theta(\log k)$ and $\Theta(k^2/\log^2 k)$. The speed-up in terms of return time is $\Theta(k)$, in both cases.

### 1.2 Related work

*Parallel random walks* As observed by Aleliunas et al. [3], the random walk is a simple randomized strategy for exploring a connected graph, visiting all its vertices within $O(n^3)$ steps in expectation, regardless of the starting location of the walker. By deploying multiple random walks in parallel, the time required to cover all vertices and edges can be further reduced. By making use of multiple random walks starting from a well-chosen initial distribution over nodes, it is possible to design fast algorithms for solving the seminal undirected s-t-connectivity problem in little memory [10,16]. More recently, multiple walks have been studied in a worst-case initialization scenario, corresponding to the setting of this paper, where the $k$ agents are placed on some set of starting nodes and deployed in parallel, in synchronous steps. The speedup of coverage of the vertices of a graph by multiple walks with respect to a single walk has been studied by Alon et al. [4], Efremenko and Reingold [14], and Elsässer and Sauerwald [15], providing a characterization of the speedup for many graph classes, in particular, random graphs in different models, and graphs with special properties, such as small mixing time compared to cover time. However, a central question posed in [4] 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.

*Deterministic graph exploration* Deterministic approaches which provide guarantees on worst-case cover time even on unknown anonymous graphs are a tempting alternative to random walks. However, their implementation proves complicated when considering anonymous networks, in which the agent is not helped by the environment, and when located at a node, it has to decide on its next move based only on its local state memory, the local port ordering at the node, and the port by which it entered the current node. It is a well-established result that no memory-less agent can explore all graphs deterministically; this impossibility result has also been extended to a finite team of memory-less agents with extended capabilities in the so-called JAG (jumping automata on graphs) model. Moreover, it has been shown [17] that an agent must be equipped with at least $|V|$ states (i.e., $\Omega(\log |V|)$ bits of memory) to be able to explore all graphs with $|V|$ nodes. On the positive side, unknown anonymous graphs can be deterministically explored by following so called universal traversal/exploration sequences. These exist for any number of nodes, and have polynomial length [3]. The cover time obtained using such an approach is, however, usually by a factor of about $|V|^2$ greater than the (expected) cover time of a corresponding random walk. It has only been shown in the last decade [24] that such universal exploration sequences can be constructed and followed using very small memory, and consequently, deterministic graph exploration can be performed by an agent with only $O(\log n)$ bits of state memory. However, the exploration time achieved by such a procedure may potentially be extremely long, expressed by a polynomial with a high exponent.

By extending the capabilities of the agent and allowing it to interact with the environment, it is possible to decrease the time of deterministic exploration without requiring the agent to use more memory. Numerous models have been proposed which rely either on the existence of informative labeling schemes in the network, or on the capability of the agent to leave pebbles on nodes, move tokens, or write to so called "white-board" memory on nodes. The reader is referred to [19] for an extensive survey.

An important line of research is devoted to *equitable strategies*, in which the environment attempts to mimic the fairness properties of the random walk with respect to the use of edges. Two such strategies, in which the agent is always directed to the least often used, or the longest unused, from among the edges adjacent to the current node were studied in [11].

When considering fairness of traversal of arcs of the graph (i.e., taking into account the direction of traversal), the strategy which directs the agent along the outgoing edge which has not been used for the longest time is precisely equivalent to the rotor-router model.

*The rotor-router model* Studies of the rotor-router started with works of Wagner et al. [26] 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 edges of the graph within $O(|V||E|)$ steps. Bhatt et al. [9] showed later that within $O(|V||E|)$ steps the agent not only covers all edges but *enters (establishes) an Eulerian cycle*. More precisely, after the initial *stabilization period* of $O(|V||E|)$ steps, the agent keeps repeating the same Eulerian cycle of the directed symmetric version $\mathbf{G}$ of graph $\mathbf{G}$ (see the model description for a definition). Subsequently, Yanovski et al. [27] and Bampas et al. [6] showed that the Eulerian cycle is in the worst case entered within $\Theta(D|E|)$ steps in a graph of diameter $D$. Considerations of specific graph classes were performed in [18]. Robustness properties of the rotor-router were further studied in [7], who considered the time required for the rotor-router to stabilise 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* [6] or *Edge Ant Walk algorithm* [26,27], and has also been described in [9] in terms of traversing a maze and marking edges with pebbles.

In the context of graph exploration, before this work, the only study of the multi-agent rotor-router was performed by Yanovski et al. [27], who showed that adding a new agent to the system cannot slow down exploration, and provided some experimental evidence showing a nearly-linear speed-up of cover time with respect to the number of agents in practical scenarios. They also show that the multi-agent rotor-router eventually visits all edges of the graph a similar number of times. Beyond this, a characterization of the behavior of the $k$-agent rotor-router in general graphs remains an open question.

*Load-balancing applications of the rotor-router* For a very large number of agents (usually $k > n$), the agents of a parallel rotor-router may be more perceived as tokens without identities, passed around between nodes. Such tokens may be seen as describing units of load on a node representing a processor in a networked system, and in this scenario, the multi-agent rotor-router mechanism has been extensively studied in the context of balancing the workload in a network. Cooper and Spencer [12] studied $d$-dimensional grid graphs, showing a constant bound on 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 [13] analyzed in more detail the distribution of tokens in the rotor-router mechanism on the 2-dimensional grid. More recently, the rotor-router has also been shown to be an effective load-balancing strategy for other regular network topologies. The case of the hypercube was studied by Akbari and Berenbrink [1], while the general case was considered by Berenbrink et al. [8].

### 1.3 Model definition

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 $\mathbf{G} = (V, \mathbf{E})$ is the directed symmetric version of $G$, having the set of arcs $\mathbf{E} = \{(v, u), (u, v) : \{v, u\} \in E\}$.

We consider the rotor-router model (on graph $G$) with $k \geq 1$ indistinguishable agents, which run in rounds, synchronized by a global clock. In each round, each agent moves in discrete steps from node to node along the arcs of graph $\mathbf{G}$. 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 $\rho_v$ is a cyclic order of the arcs (in graph $\mathbf{G}$) outgoing from node $v$, $\pi_v$ is an outgoing arc 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 $\rho_v$ of the arcs outgoing from $v$ is fixed at the beginning of exploration and does not change in any way from step to step (unless an edge is dynamically added or deleted as discussed in the previous section). For an arc $(v, u)$, let $next(v, u)$ denote the next arc after arc $(v, u)$ in the cyclic order $\rho_v$.

The exploration starts from some initial configuration and then keeps running in all future rounds, without ever terminating. During the current round, first each agent $i$ is moved from node $r_i$ traversing the arc $\pi_{r_i}$, and then the port pointer $\pi_{r_i}$ at node $r_i$ is advanced to the next arc outgoing from $r_i$ (that is, $\pi_{r_i}$ 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 round 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 round, then it will send one of the agents along the arc $\pi_v$, and the other along the arc $(v, next(\pi_v))$. In some considerations, we will also assign explicit labels $\{0, 1, \ldots, \deg(v) - 1\}$ to the ports adjacent to $v$, in such a way that initially the label of arc $\pi_v$ is 0, and if the label of $(v, u)$ is $i$ then the label of $next(v, u)$ is $(i + 1) \mod \deg(v)$. Then, at the completion of any round, the total number of traversals of agents along an arc $(v, u)$ is equal to $\left\lceil \frac{e_v - port_v(u)}{\deg(v)} \right\rceil$, where $e_v$ is the total number of times agents exited node $v$ until the completion of the round and $port_u(v)$ denotes the label of the port leading from $v$ to $u$.

In all our considerations, we will assume that the initialization of ports and pointers in the system is performed by an adversary. In particular, when studying a best-case scenario of initial agent locations, we assume that the ports and pointers have been set by the adversary so as to maximize the studied parameter (e.g., cover time). For the case of the ring, there exists only one cyclic permutation of the two neighbors of each node, hence only the initial pointer arrangement (and not the configuration of ports) is relevant.

## 2 Techniques for the multi-agent rotor-router

### 2.1 Delayed deployments

In our work we will consider both the unmodified $k$-agent rotor-router system $R[k]$ and its *delayed deployments*, in which some agents may be stopped at a node, skipping their move for some number of rounds. A delayed deployment $D$ of $k$ agents is formally defined as a function $D : V \times \mathbb{N} \to \mathbb{N}$, where $D(v,t) \geq 0$ represents the number of agents which are stopped in vertex $v$ in round $t$ of the execution of the system. A deployment is delayed if $D(v,t) > 0$ for at least one pair $v, t$. For the undelayed rotor-router system $R[k]$ we have $R[k](v,t) = 0$, for all $v$ and $t$. Delayed deployments may be conveniently viewed as algorithmic procedures for delaying agents, and are introduced for purposes of analysis, only.

We will say that a node $v$ is *visited* by an agent in round $t$ if the agent traverses an edge incoming to $v$ during step $t$. The agent is then located at $v$ at the start of round $t+1$. Let $n_v^D(t)$ denote the total number of visits of agents to node $v$ during the interval of rounds $[1,t]$ for agents following some (possibly delayed) deployment $D$, and let $C(D)$ be the cover time of this deployment. The notation $n_v^D(0)$ refers to the number of agents at a node directly after initialization (at the start of round 1).

We start by showing that by delaying more agents in a deployment, one cannot increase the number of visits to nodes at any time. We assume that all considered deployments start from the same (arbitrarily chosen) initial configuration.

**Lemma 1** *Let $D_1$ and $D_2$ be two (possibly delayed) deployments of the $k$-agent rotor-router system, such that for all vertices $v \in V$ and rounds $t$, $D_1(v,t) \geq D_2(v,t)$. Then, for all vertices $v \in V$ and rounds $t$, we have $n_v^{D_1}(t) \leq n_v^{D_2}(t)$.*

*Proof* For $t = 0$, the claim holds, since by definition:

$$n_v^{D_1}(0) = n_v^{D_2}(0) = n_v^{R[k]}(0), \quad \text{for all } v \in V. \tag{1}$$

Now, denote by $e_v^{D_1}(t)$ and $e_v^{D_2}(t)$ the total number of traversals of arcs outgoing from $v$ during the interval of rounds $[1,t]$ for executions $D_1$ and $D_2$, respectively. For an arbitrary agent, the difference between the number of times the agent leaves $v$ in rounds $[1,t+1]$ and the number of times it enters node $v$ in rounds $[0,t]$ is equal to either $-1$ or 0, depending on whether the agent is delayed at $v$ in round $t+1$ or not. Summing over all agents, we obtain:

$$e_v^{D_i}(t+1) = n_v^{D_i}(t) - D_i(v,t+1), \quad i \in \{1,2\} \tag{2}$$

The rest of the proof proceeds by induction on time $t$. Suppose that for some $t > 1$, $n_v^{D_1}(t-1) \leq n_v^{D_2}(t-1)$ holds for all $v \in V$. Then, we have from (2):

$$e_v^{D_1}(t) + D_1(v,t) \leq e_v^{D_2}(t) + D_2(v,t)$$

and since $D_1(v,t) \geq D_2(v,t)$:

$$e_v^{D_1}(t) \leq e_v^{D_2}(t), \quad \text{for all } v \in V.$$

Now, fix an arbitrary node $u$ and observe that the number of visits to node $u$ within the interval $[1, t+1]$ is equal to the sum of the number of agents placed at $u$ in round 1, and the number of times an agent exited one of its neighbors $v \in \Gamma(u)$ along an arc $(v,u)$ in rounds $[1,t]$:

$$n_u^{D_i}(t+1) = n_u^{D_i}(0) + \sum_{v \in \Gamma(u)} \left\lceil \frac{e_v^{D_i}(t) - port_v(u)}{\deg(v)} \right\rceil, \tag{3}$$

where we took into account that agents leaving a node $v$ exit along the ports adjacent to $v$ in round-robin fashion. Combining equations (1), (2), and (3) we obtain $n_u^{D_1}(t+1) \leq n_u^{D_2}(t+1)$. Since $u \in V$ was arbitrarily chosen, the inductive claim follows. □

We remark that the above lemma immediately implies that for undelayed deployments $R[k-1]$ and $R[k]$ with identical initial positions of $k-1$ agents we have $n_v^{R[k-1]}(t) \leq n_v^{R[k]}(t)$, since the $(k-1)$-agent rotor-router $R[k-1]$ is equivalent to a deployment of the $k$-agent rotor-router with one agent permanently stopped. (This observation is due to [27].)

**Lemma 2** *Let $D$ be a delayed deployment of the $k$-agent rotor-router system. Let $T$ be any fixed time round, and let $\tau$ be the number of rounds in the interval $[1,T]$ such that all the agents are active in $D$, i.e., $\tau = |\{t \in [1,T] : \forall_{v \in V}\ D(v,t) = 0\}|$. Then, for all vertices $v$, we have: $n_v^{R[k]}(\tau) \leq n_v^D(T) \leq n_v^{R[k]}(T)$.*

*Proof* The right inequality follows directly from Lemma 1. To prove the left inequality, we rewrite for round $t \geq 1$ the sets of recurrence equations (2) and (3) on the number of visits and exits to each node $v$ for deployment $D$:

$$\begin{cases} e_v^D(t) = n_v^D(t-1) - D(v,t), \\ n_v^D(t) = n_v^D(0) + \sum_{w \in \Gamma(v)} \left\lceil \frac{e_w^D(t-1) - port_w(v)}{\deg(w)} \right\rceil, \\ n_v^D(0) = n_v^{R[k]}(0), \quad e_v^D(0) = 0. \end{cases}$$

Consider a function $f : [1,\tau] \to [1,T]$, with $f(i)$ being the $i$th time round in which all agents are active in delayed deployment $D$. Denote by $F \subseteq [1,T]$ the image of $f$. Taking into account that $D(v,t) = 0$ for all $t \in F$ and that the counters $e_v^D$ and $n_v^D$ are always non-decreasing in time, we obtain the following set of inequalities by restricting evolution to moments of time $t = f(i)$, with $i \in [1,\tau]$:

$$\begin{cases} e_v^D(f(i)) = n_v^D(f(i)-1) - D(v, f(i)) \\ \qquad\qquad \geq n_v^D(f(i-1)) - 0 = n_v^D(f(i-1)), \\ n_v^D(f(i)) = n_v^D(0) + \sum_{w\in\Gamma(v)} \left\lceil \frac{e_w^D(f(i)-1) - port_w(v)}{\deg(w)} \right\rceil \\ \qquad\qquad \geq n_v^D(f(0)) + \sum_{w\in\Gamma(v)} \left\lceil \frac{e_w^D(f(i-1)) - port_w(v)}{\deg(w)} \right\rceil, \\ n_v^D(f(0)) = n_v^{R[k]}(f(0)), \quad e_v^D(f(0)) = 0, \end{cases}$$

where we put $f(0) = 0$ for convenience of notation. By comparing the above with the corresponding equations for the undelayed rotor-router $R[k]$, written for round $i \in [1, \tau]$:

$$\begin{cases} e_v^{R[k]}(i) = n_v^{R[k]}(i-1), \\ n_v^{R[k]}(i) = n_v^{R[k]}(0) + \sum_{w\in\Gamma(v)} \left\lceil \frac{e_w^{R[k]}(i-1) - port_w(v)}{\deg(w)} \right\rceil, \\ e_v^{R[k]}(0) = 0. \end{cases}$$

it follows by induction that $n_v^D(f(i)) \geq n_v^{R[k]}(i)$. Putting $i = \tau$, we obtain the sought inequality $n_v^D(T) \geq n_v^{R[k]}(\tau)$. □

Observe that by the above lemma, we have that if node $v$ is visited for the first time after $T$ rounds in a delayed deployment $D$, i.e., $n_v^D(T) = 0$ and $n_v^D(T+1) \geq 1$, then $n_v^{R[k]}(\tau) = 0$ and $n_v^{R[k]}(T+1) \geq 1$. From this, we directly obtain the key lemma for the approach we use to analyzing the cover time of $k$-rotor-router systems in this paper.

**Lemma 3** (the slow-down lemma) *Let $R[k]$ be a $k$-rotor router system with an arbitrarily chosen initialization, and let $D$ be any delayed deployment of $R[k]$. Suppose that deployment $D$ covers all the vertices of the graph after $T = C(D)$ rounds, and in at least $\tau$ of these rounds, all agents were active in $D$. Then, the cover time $C(R[k])$ of the system can be bounded by:*

$$\tau \leq C(R[k]) \leq T.$$

□

If the deployment $D$ is defined so that agents in $D$ are delayed in at most a constant proportion of the first $C(D)$ rounds, then the above inequalities lead to an asymptotic bound on the value of the undelayed rotor-router, $C(R[k]) = \Theta(C(D))$. This is the case, e.g., in the proof of Theorem 1.

### 2.2 Agent domains on the ring

When multiple agents are deployed on a ring, all the visited nodes are partitioned between the agents into so-called *domains*. Informally, the domain of each agent is a sub-path consisting of the nodes for which the agent was the last agent visiting that node. We will later observe that agents on the ring are patrolling their respective domain, by moving from one endpoint to another. Our goal in this section will be to show that after sufficiently large number of steps, each domain will have size close to $n/k$. We would like also to show that domains do not shrink too much during exploration, i.e., if at some step $t$ each domain is of size at least $x$, then for any $t' > t$, each domain is of size at least $x/2$. The latter result will be shown assuming that the pointers are initialized *negatively*, i.e., in the scenario where during the first visit to any vertex by some agent, this agent is directed back to its previous location.

*Ordering of visits to nodes* Let $t$ be an arbitrarily fixed time step, and let $v$ be a vertex of the ring that is not visited by an agent at the beginning of step $t$, but has been visited in some time step before $t$. Consider the position of pointer at vertex $v$ at time $t$. Directly after the last visit to vertex $v$ before $t$, at least one agent exited $v$ in *the other* direction, and then the pointer was advanced to its current position. By $o(v,t)$ we denote the first node of the ring in the direction opposite to the one indicated by the pointer at $v$ at the beginning of step $t$, such that $o(v,t)$ contains an agent at the beginning of time step $t$. If $v$ has not been visited until time $t$, we define $o(v,t) = \perp$. For any vertex $v$ that contains an agent at the beginning of step $t$, we put $o(v,t) = v$. Denote by $P(v,t)$ the directed path from $v$ to $o(v,t)$, for all $v, t$ for which $o(v,t) \neq \perp$.

**Lemma 4** *For any $v, t$ for which $o(v,t) \neq \perp$*

1. *node $o(v,t)$ is the position of agent $\alpha$ which was the last (or one of the last) to visit $v$ up to time $t$ (at the beginning of some step $t' \leq t$),*
2. *during steps $t', t'+1, \ldots, t-1$, agent $\alpha$ walked directly along the path $P(v,t)$ from $v$ to $o(v,t)$,*
3. *for any node $v'$ on path $P(v,t)$, we have $o(v,t) = o(v',t)$,*
4. *each node on path $P(v,t)$ was visited exactly once in the range of time steps $t'$ and $t$.*

*Proof* If $v$ contains an agent then $v = o(v,t)$ and the claim is trivial. We will show the lemma by induction on the number of edges of $P(v,t)$. Observe that if a node $v$ has been last visited in time $t' < t$ by only one agent then the direction opposite to the one indicated by the pointer at $v$ indicates the direction followed by $\alpha$ during step $t'$. Note that it can happen that in $t' < t$ node $v$ was visited by more than one agent. Then both directions lead to an agent that was the last to visit $v$ and then by $\alpha$ we denote the one that is in $o(v,t)$.

During step $t'$, agent $\alpha$ moved to a node $v^*$, and $v^*$ is the second node on the path $P(v,t)$. If $P(v,t)$ has one edge then $v^* = o(v,t)$, and the lemma holds. Assume that the lemma holds for all $v'$ for which $|P(v',t)| < x$, for any $x \geq 2$. Take any node $v$ such that $|P(v,t)| = x$. Let $v'$ be the second node on $P(v,t)$. Since $x \geq 2$ then $v'$ does not contain an agent in

step $t$ and did contain any agent in step $t'+1$, thus $t'+1 < t$. Moreover, since $t'$ was the last visit to $v$ before time $t$, during steps $t', t'+1, \ldots, t-2$, the agent did not move to $v$. Thus, $v'$ was visited exactly once in this interval, since otherwise, by the properties of the rotor-router, the agent would have continued to $v$, visiting $v$. Thus, $t'+1$ was the last visit to $v'$ before time $t$. Moreover, in step $t$, the directions of the pointers at $v$ and $v'$ are the same, thus $o(v,t) = o(v',t)$. We can use the inductive assumption for $v'$ to complete the proof of the claim for $v$. □

**Lemma 5** *In a $k$-agent rotor-router system, if at the beginning of some time step $t_0$ at most 2 agents are located at each node of the ring, then in any subsequent time step $t \geq t_0$ at most 2 agents are located at each node of the ring.*

*Proof* Clearly, it is sufficient to show the claim for $t = t_0+1$. Since at the beginning of step $t_0$ each node hosts at most 2 agents and since each node of the ring is of degree 2, each edge is traversed by at most 1 agent during step $t_0$. Thus, at the beginning of step $t_0+1$ each node hosts at most 2 agents. □

*Notion of a domain* We now proceed to formally define the *domain* $V_a(t)$ of each agent $a$, for any moment of time $t$, such that at time $t$, each vertex on the ring contains at most 2 agents. Lemma 4 shows that nodes $v$ with the same value of $o(v,t)$ (different from $\perp$) form sub-paths of the ring. If agent $\alpha$ is the only agent to occupy a node $v^*$, then the sub-path of nodes with $o(v,t) = v^*$ will define its domain $V_\alpha(t)$:

$$V_\alpha(t) = \{v : o(v,t) = v^*\}.$$

However, if a node contains two agents, we will divide the sub-path between the agents, forming their domains. Formally, let node $v^*$ contain two agents $a$ and $b$ in step $t$, and let the pointer at $v^*$ point in the clockwise direction. Then, the domains of $a$ and $b$ are defined as follows:

$$\begin{aligned} V_a(t) = \{&v : o(v,t) = v^* \\ &\text{and } v \text{ is in the anticlockwise direction from } v^*\} \\ &\cup \{v^*\} \\ V_b(t) = \{&v : o(v,t) = v^* \\ &\text{and } v \text{ is in the clockwise direction from } v^*\} \end{aligned}$$

If the pointer at $v^*$ points in the anticlockwise direction, then the domains of $a$ and $b$ are defined analogously, but vertex $v^*$ is added to domain $V_b(t)$. Observe that agents are indistinguishable and the assignment of clockwise and anticlockwise parts to the agents is arbitrary. For the remaining, unvisited nodes we define a dummy domain $V_\perp(t) = \{v : o(v,t) = \perp\}$ with no agent in it.

Note that by Lemma 5, we know that if agent domains are well defined for some time $t_0$, then they are also well defined for any $t \geq t_0$. We will later show that such $t_0$ indeed exists.

If we consider the evolution of the set of domains in time, we can observe that an agent $b$ performing a traversal of its domain $V_b$ will always attempt to extend the path of its domain, capturing one node of both the neighboring domains. If agent $b$ is neighboring the unvisited region $V_\perp$, it will capture one node per traversal if the pointers are initialized negatively. In the case of differently initialized pointers it may capture more nodes. Thus, some nodes will frequently change their membership in domains. However, if two neighboring domains $V_a$, $V_b$ are of similar size, then the frequency of visits to the extremal nodes of the respective domains by each of two agents $a$, $b$ is similar. In such a case, the border between domains $V_a$ and $V_b$ oscillates by 1 node in each traversal of the agents made within their respective domains. To make further analysis easier, we introduce an additional concept of *lazy domains* which will be insensitive to such oscillations. The lazy domain of each agent will be defined as a subset of its domain restricted to vertices satisfying a specific condition, describing its "interior". Unvisited nodes from $V_\perp$ will not belong to any lazy domain.

We say that a visit by agent $a$ in step $t$ to a vertex $v$ is a *propagation* if during step $t+1$, agent $a$ moves to a vertex that is different than the one from which it entered $v$ during step $t$. A visit after which the agent moves back to the vertex from which it previously came is called a *reflection*.

**Definition 1** For any time $t$, the *lazy domain* $V'_a(t)$ of agent $a$ is defined as a subset of its domain $V_a(t)$ containing only such vertices $v$ that in the step $t' < t$ when $v$ was last visited, $v$ was visited by one agent only and this visit was a propagation.

**Lemma 6** *If $k > 1$ then for any time $t$ and any agent $a$:*

1. *set $V_a(t)$ induces a sub-path of the ring,*
2. *set $V'_a(t)$ induces a sub-path containing all nodes of $V_a(t)$ except possibly its endpoints.*

*Proof* Let $v^*$ be the location of agent $a$ at the end of step $t$.

First, consider the case when there are two agents at node $v^*$. By Lemma 4 and the definition of $V_a(t)$, Claim 1 holds. Assume without loss of generality that $V_a(t)$ contains nodes in the anticlockwise direction from $v^*$ and let $v \in V_a(t)$ be the farthest node from $v^*$ in the anticlockwise direction. By Lemma 4 one of the agents located at $v^*$ walked directly from $v$ to $v^*$ until step $t$. Thus, all the last visits in each of the vertices of the path $P(v,t)$ were propagations, except possibly for the vertices at the endpoints of this path. This shows Claim 2.

Now assume that there is exactly one agent in $v^*$. Again, Claim 1 holds by the definition of domains and Lemma 4. To show Claim 2, we need to prove that if $v^*$ is not the

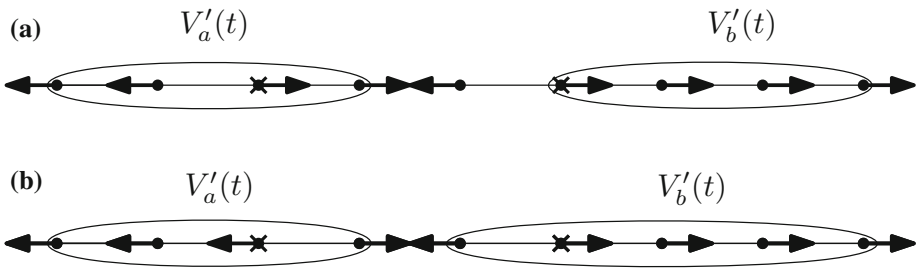


**Fig. 1** Illustrations of different types of borders. Positions of the two agents are indicated with crosses. **a** Node-type border. **b** Edge-type border

endpoint of the sub-path $V_a(t)$, then the visit to $v^*$ in step $t$ is a propagation. Denote by $v_1$ and $v_2$ the endpoints of $V_a(t)$. Denote by $t_1, t_2$ the times of last visits to $v_1$ and $v_2$ before $t$ and assume that $t_1 < t_2$. By Lemma 4, agent $a$ walked directly to $v^*$ after visiting $v_1$. Since it visited $v_2$ after $t_1$ and before $t$, it had to visit $v^*$ and at least one visit to $v^*$ in this time interval was a propagation. But by Lemma 4, agent $a$ visited $v^*$ at most once between steps $t_1$ and $t-1$. Since this unique visit was a propagation, we have $v^* \in V'_a(t)$. Next, by Lemma 4 we have that all nodes of the paths $P(v_1, t)$ and $P(v_2, t)$, except their endpoints, belong to $V'_a(t)$. Since $v^*$ is the common endpoint of these paths, we have that $V'_a(t)$ is a sub-path of the ring. □

Let $a$ and $b$ be two agents with consecutive lazy domains $V'_a(t)$ and $V'_b(t)$. We will say that agent $a$ *moves the border* between $a$ and $b$ at time $t$ if $t$ is a time such that for some node $v$ we have $v \in V'_a(t)$ and there exists $t' < t$ such that $v \in V'_b(t')$ and in steps $t'+1, \ldots, t-1$ node $v$ does not belong to the lazy domain of any agent. In this situation, we will say that $a$ moves the border by *capturing* node $v$.

If the lazy domains of agents $a$ and $b$ are nonempty, then we introduce the following notation for the two possible types of border between them:

- *Vertex-type border.* In this case there is a vertex between the lazy domains of agents $a$ and $b$ (see Fig. 1a).
- *Edge-type border.* In this case the two extremal nodes from lazy domains $V'_a$ and $V'_b$ are neighbors (see Fig. 1b). Observe that in this case, since both endpoints of the edge on the domain border have been visited in the propagation type and belong to different lazy domains, this edge was traversed in both direction by both agents during the same step. This can be seen as a swap of the agents on the edge. Since the agents are indistinguishable, then the states at this edge before and after the swap are identical.

Observe that the lazy domain border might consists of two nodes only in one special case, when the edge between the domains has just been traversed for the first time. If an edge $e$ is traversed in step $t$ then in step $t+1$ or $t+2$, one of its endpoints will be visited in a propagation move (this will happen in step $t+2$ if the agent traversed $e$ again in $t+1$). Thus, if two neighboring domains are nonempty and each edge within (and between) the domains has been traversed, then the border is either a vertex or an edge.

We say that an agent *visits an edge-type border* in step $t$ if during step $t$ it traverses the edge during step $t$. We say that an agent *visits a node-type border* in step $t$ if it is located in the node at the end of step $t$.

Given the above notation, in the following we will describe the structure of the move of an agent within its domain. We want to show that in order to move a border, the same agent has to visit the border twice in a row. Secondly we will show that between two visits to the same border, the agent visits all the nodes from its lazy domain twice. We will later use these facts to show that the domains will eventually even out in time.

**Proposition 1** *If at time $t$, agent $a$ moves the border between $a$ and $b$ by traversing an edge $e = (v_a, v_b)$ and capturing node $v_b$, where $v_b \in V'_b(t)$, then if $t' < t-1$ was the time of the previous visit of $a$ to $v_a$ then in steps $t'+1, t'+1, \ldots, t-1$ node $v_a$ was not visited by $b$.*

*Proof* As a direct consequence of the rules of the rotor-router, if two consecutive visits to a node of the ring are from different directions then these visits are of the same type (either two reflections or two propagations). On the other hand if the visits are from the same direction then they are of different types.

First, assume that agent $a$ is moving a node-type border when capturing node $v_b$. Since $v_a$ is the extremal node, then the previous visit of agent $a$ to this vertex was a reflection. Since the current visit is a propagation, it follows that between the two visits by agent $a$ there was no visit by $b$.

On the other hand, consider the case when $a$ moves an edge-type border by traversing it during step $t$. Then, at the beginning of step $t$, agent $a$ is located at $v_a$ and the border is of the edge type. This means that the last traversal of edge $(v_a, v_b)$ was a swap between the agents in step $t'$. Thus, node $v_a$ was not visited between steps $t'+1$ and $t-2$. Clearly in step $t-1$ only one agent visits $v_a$ since otherwise by the principles of the rotor-router, in step $t$ agents would move in the opposite direction and we know that $a$ traversed edge $e$. □

After agent $a$ has captured a vertex from neighboring lazy domain, it is directed back towards its own domain. Thus, the border always moves by one vertex. If the border was an edge, then the move of the border consists in agent $a$ visiting the endpoint of the edge belonging to the domain of $b$. In this case the lazy domain of $b$ shrinks by one vertex and the border becomes a vertex. It can also happen that an edge-type border becomes a vertex-type border. In all cases, the border is always moved to an adjacent vertex or edge and the agent is directed back towards its domain. Finally, if agent $a$ is adjacent to an unexplored region, then, since, we assume that all pointers are initialized negatively, the agent can capture only one new vertex and it will be directed back towards its domain.

In the following we will show how long it takes for an agent to return to the same border.

**Proposition 2** *For an agent $a$, let $t_1$ and $t_2$ be the times of two consecutive visits of $a$ to a fixed (left or right) border of its lazy domain. Then, in the time interval $[t_1, t_2]$, agent $a$ visits all the nodes from its lazy domain $V'_a(t_2)$ exactly twice.*

*Proof* After the visit to border $B_1$ in step $t_1$, the agent is directed towards its lazy domain. Observe that the pointers of its lazy domain will guide the agent towards the other border $B_2$. Indeed, pointers at each node $v \in V'_a(t)$ point away from the current location of agent $a$. Since by Lemma 6, $V'_a(t)$ induces a sub-path of the ring and since the previous visit to each of vertices from $V'_a(t)$ directed $a$ towards border $B_1$, then the agent will walk towards $B_2$ in step $t_1+1, t_1+2, \ldots$. Eventually in step $t^*-1$, agent $a$ will reach the other border $B_2$. If the agent moves the border $B_2$, let $t^*$ be the first step during which the agent changes its direction. This shows that between the visit of the agent to $B_1$ in step $t_1$ and its visit to $B_2$ in step $t^*$, the agent visits the whole lazy border $V'_a(t^*)$. Symmetrically, between visits to $B_2$ in step $t^*$ and to $B_1$ in step $t_2$, the agent visits the whole lazy border $V'_a(t_2)$. Observe that $V'_a(t^*) \subseteq V'_a(t_2)$ since in steps $t^*, t^*+1, \ldots, t_2$ agent $a$ walks within its domain and cannot move any of its borders. □

Observe that if the borders of $V'_a$ do not move within the time interval $[t_1, t_2]$, then $a$ visits all the nodes from $V'_a(t_1)$ exactly twice.

**Lemma 7** *Let $t$ be a time step such that an agent $a$ moves the border between its domain and that of agent $b$. If $t^* < t$ was the previous time step when $a$ visited the border between the domains of $a$ and $b$, and if at time $t^*$ all unvisited nodes of the ring had negatively initialized pointers, then $2|V'_a(t)| \leq t - t^* \leq 2|V'_b(t^*)| + 3$.*

*Proof* The left inequality is implied by Proposition 2. To show the right inequality, observe that agent $b$ cannot visit the border between $a$ and $b$ in the time interval $[t^*, t-1]$, as otherwise the move would not be possible by Proposition 1. Now, consider the trajectory of agent $b$ starting from step $t^*$. It can move its other border (i.e., the border of the domain of $b$ which is not a border of the domain of $a$) at most once before getting back to the border with $a$. Thus, agent $b$, regardless of its position in step $t^*$, after $2|V'_b(t^*)| + 4$ steps will have returned to the border with $a$. □

Denote by $min(t)$ the minimum size of a lazy domain $V'_a(t)$ over all agents $a$ in step $t$. In the following lemma we will show that if at some point of time $t_0$ all lazy domains have size at least $20k$, for a sufficiently large integer $k$, then the domains will never degenerate and in every $t > t_0$ each lazy domain will have size at least $15k + 5$. (All the missing proofs are deferred to the "Appendix".)

**Lemma 8** *Suppose that $k \geq 6$. Let $t_0$ be a time step such that at time $t_0$ all unvisited nodes of the ring have negatively initialized pointers, at time $t_0$ each node contains at most 2 agents and each lazy domain has size at least $\mu$ nodes, where $\mu \geq 10k$. Then, for any $t \geq t_0$:*

1. *if $a, b$ are any two agents with consecutive domains and if $|V'_b(t)| + 8 \leq |V'_a(t)|$ and $|V'_b(t)| \leq \mu - 3$, then in step $t$, the border between domains $V_a$ and $V_b$ is not moved by agent $a$,*
2. *$min(t) \geq \mu - 5k + 2$.*

**Lemma 9** *Assume that $k \geq 6$ and that in step $t$ all lazy domains have size at least $20k$. If for some three agents $a, b, c$ occupying consecutive domains we have $|V'_b(t)|+8 \leq |V'_a(t)|$ and $|V'_b(t)| \leq 2|V'_c(t)|$, then in step $t$, the border between lazy domains $V'_a$ and $V'_b$ is not moved by agent $a$.*

**Lemma 10** *Assume $k \geq 6$ and that at some moment of time $t_0$, each lazy domain has size at least $20k$. Then, for any $t \geq t_0$, if for some two agents $a, b$ with adjacent domains we have $|V'_a(t)| > 1.1|V'_b(t)|$, then in step $t+1$ the border between $a$ and $b$ will not be moved by agent $a$.*

*Proof* Assume, by contradiction, that agent $a$ moves the border with $b$ in step $t+1$. Let $t^*$ be the last time step when agent $b$ visited its other border. By Lemma 7, we have $2|V'_a(t)| \leq t - t^* \leq 2|V'_b(t^*)| + 3$. Note that it may be the case that $|V'_b(t^*)| > |V'_b(t)|$, if agent $b$ lost some nodes of its lazy domain during the time interval $[t^*, t]$; however, we can bound the number of nodes lost. Since the size of every domain is at least $15k + 5$, by Lemma 8, agent $b$ loses at most 1 node once every $30k + 10$ time steps. Thus, during $2|V'_b(t^*)| + 3$ time steps, $b$ lost at most $\frac{2|V'_b(t^*)|+3}{30k} + 1$ nodes of its lazy domain. Thus

$$|V_b'(t)| \geq |V_b'(t^*)| \left(1 - \frac{1}{15k}\right) - \frac{1}{10k} - 1,$$
$$|V_b'(t^*)| \leq \left(|V_b'(t)| + \frac{1}{10k} + 1\right)\left(1 + \frac{1}{15k-1}\right).$$

Since $k \geq 6$, and $|V_b'(t)| \geq 15k + 5 \geq 95$, then $|V_b'(t^*)| \leq |V_b'(t)| \left(1 + \frac{1}{10k|V_b'(t)|} + \frac{1}{|V_b'(t)|}\right)\left(1 + \frac{1}{15k-1}\right) < 1.08|V_b'(t)|$. Thus

$$t - t^* \leq 2|V_b'(t^*)| + 3 < 2.16|V_b'(t)| + 3 < 2|V_a'(t)| \leq t - t^*,$$

a contradiction. □

**Lemma 11** *Consider an interval $I = [t, t + 2n^2 - 1]$ of $2n^2$ consecutive time steps. For any two agents $a$, $b$ with adjacent domains, if the relation $|V_a'(t^*)| - 4 > |V_b'(t^*)|$ holds throughout interval $I$ for all time steps $t^* = t, t + 1, t + 2, \ldots, t + n^2 - 1$, then the border between $a$ and $b$ will be moved by $b$ for at least one time step in $I$.*

*Proof* Starting from any $t^* \in I$, agent $a$ takes at least $2|V_a'(t^*)|$ steps to traverse its entire domain in both directions. Agent $b$ takes at most $2|V_b'(t^*)| + 4$ steps to perform a similar cyclic traversal, visiting its whole lazy domain twice and both its borders. Thus, if $2|V_a'(t^*)| > 2|V_b'(t^*)| + 8$, then the cycle of $b$ is shorter by at least four steps. Eventually, after a sufficiently large number of time steps (after at most $2n^2$ steps counting from the beginning of interval $I$), agent $b$ will visit the border between the domains of $a$ and $b$ twice in some time interval $[t_1, t_2]$ and $a$ will not visit the border in this time interval. Thus, $b$ will move the border towards $a$, gaining one node. □

From our considerations, we finally obtain a lemma which will prove crucial in characterizing the limit behavior of the rotor-router on the ring.

**Lemma 12** (agent domains) *Let $k \geq 6$. If at some time step $t$ every lazy domain has size at least $20k$ and the unexplored part of the ring has negatively initialized pointers, then after a sufficiently large number of steps, the sizes of adjacent lazy domains will differ by at most 10.*

*Proof* Clearly if at step $t$, each domain has size at least $20k$ then no 3 agents occupy the same node. We know by Lemma 8 that if initially every lazy domain has size at least $20k$, then after any number of steps, every domain will have size at least $15k + 2$. If $a$ and $b$ are neighbors and at time $t^*$, we have $|V_a'(t^*)| \geq 2|V_b'(t^*)|$, then we will say that there is a *significant difference* between $a$ and $b$. If there is a significant difference between two adjacent domains of $a$ and $b$, then by Lemma 10, the border will never move towards the smaller domain and by Lemma 11, the border will eventually move towards the bigger domain. Thus, significant differences will eventually disappear. Now, if there is no significant difference between sizes of any neighboring domains, then by Lemma 9, the border can move in the wrong direction (towards the smaller domain) only if the difference is at most 8. Observe that if agent $b$ is neighboring the unexplored region, then since the pointers are initialized negatively, then we can view it as a domain with infinite size and Lemma 9 also holds in this case.

On the other hand, by Lemma 11, if the difference between adjacent lazy domains is at least 4 for a sufficiently large number of consecutive time steps, then the border will move towards the bigger domain. Thus, if the difference between the sizes of two adjacent lazy domains is at least 8, then the border cannot move in the wrong direction and will eventually move in the correct direction. Thus, we obtain that the sizes of adjacent lazy domains will eventually differ by at most 10. □

### 2.3 Continuous-time approximation

To provide an asymptotic description of the behavior of agent domains in time, we introduce the continuous-time approximation of the agents' behavior. This is useful under the assumption that the sizes of all the domains are sufficiently large, i.e., that the change of size of $V_i(t)$ in the number of rounds of the order $|V_i(t)|$ is negligible with respect to $|V_i(t)|$.

Suppose that the domains of the agents are ordered along the ring as $V_0(t), V_1(t), \ldots, V_k(t)$, where $V_0(t)$ denotes the set of unvisited vertices. Now, the $i$th agent is defined as the agent whose domain is $V_i(t)$. Assuming that only the $i$th agent is moving, the agent will reach each of the endpoints of its domain every $1/(2|V_i(t)|)$ rounds. Consequently, within $T$ rounds, the agent enlarges its domain by approximately $T/(2|V_i(t)|)$ to the left, and $T/(2|V_i(t)|)$ to the right, thus by about $T/|V_i(t)|$ in total. This movement is counteracted by the moves of the adjacent agents occupying domains $V_{i-1}$ and $V_{i+1}$. Consequently, we define the *continuous-time approximation* of the rotor-router through the set of differential equations:

$$\frac{dv_i(t)}{dt} = \frac{1}{v_i(t)} - \frac{1}{2v_{i-1}(t)} - \frac{1}{2v_{i+1}(t)}, \quad \text{for } 1 \leq i \leq k,$$

where $v_i(t) = |V_i(t)|$, for all $1 \leq i \leq k$. The interpretation of $v_0(t)$ and $v_{k+1}(t)$ depends on whether the whole ring has already been covered: if so, then $v_{k+1}(t) \equiv v_1(t)$ and $v_0(t) \equiv v_k(t)$, since then domains of agents 1 and $k$ are adjacent. If not, i.e., if $|V_0(t)| > 0$, then we put $v_0(t) = v_{k+1}(t) = +\infty$. The latter assumption corresponds to a barrier of negatively initialized pointers present beyond both endpoints of the explored portion of the ring, and allows us to consider the worst-case time of first exploration of the

ring (cf. e.g. Lemma 13 for a corresponding initialization of parameters in the discrete setting).

The rough intuition behind the analysis of the continuous model is the following. In the analysis, we postulate a separation of variables $i$ and $t$ in the definition of $i$, as follows: $v_i(t) = f(t)/g_i$, for some functions $f : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{N} \to \mathbb{R}$. When the domains have reached their eventual lengths after the ring has been covered, we expect to have $\frac{dv_i(t)}{dt} = 0$, and so $g_i = (g_{i-1} + g_{i+1})/2$, for all domains $i$ (where we assume $g_{k+1} = g_1$ and $g_0 = g_k$ by the cyclic condition). This suggests that in this case $g$ is a constant function, and the domain size is identical for all agents. On the other hand, for the case when the ring has not yet been covered suggests solutions of the form: $\frac{df(t)}{dt} \sim \frac{1}{f(t)}$ and $g_{i+1} = 2g_i - g_{i-1} - \frac{1}{g_i}$. The time dependency of $f$ unravels as $f(t) \sim \sqrt{t}$, while the dependency of $g$ on $i$ takes a more complicated form (the analytical solution is linked to the inverse normal error function), but by dropping the less relevant term $(-\frac{1}{g_i})$ from the formula, we obtain an asymptotic solution of the form $g(i) \sim \Theta(i)$. Thus, overall, we expect domain sizes to grow with the square root of time, with the relative size of the domain of the $i$th agent being inversely proportional to the value of $i$.

Since the continuous differential model is not directly applicable to our considerations, we do not provide a formal analysis of its properties. In fact, this model provides the basic intuition for many of the proofs, but the main difficulty lies in taking into account the differences between the continuous-time model and the real rotor-router. In particular, we have to consider the position of the agent within its domain, the discrete changes of the domain size in time, and the initial pointer arrangement in the unvisited part of the ring.

## 3 Cover time of the multi-agent rotor router on the ring

### 3.1 Worst-case initial placement

The following lemma introduces a sequence $\{a_i\}_{i=0}^{k+1}$, useful in analyzing initial placements in which all agents start from the same point of the ring. It corresponds to a normalized solution to the continuous-time model of the rotor-router (i.e., $a_i(t) = v_i(t)/\sum_j v_j(t)$), subject to the constraint that the proportions of domain sizes do not change in time (i.e., $\frac{da_i(t)}{dt} = 0$), and specific boundary conditions.

**Lemma 13** *For any $k > 3$ there exists a sequence $(a_0, a_1, \ldots, a_k, a_{k+1})$, where $a_i \in \mathbb{R}_+ \cup \{\infty\}$ which satisfies the following properties:*

*(1)* $a_0 = +\infty$,
*(2)* $a_{k+1} = a_k < a_{k-1} < \cdots < a_1$,
*(3)* $\sum_{i=1}^{k} a_i = 1$,
*(4)* $a_i \cdot a_1 = \frac{2}{a_i} - \frac{1}{a_{i-1}} - \frac{1}{a_{i+1}}$, *for all* $1 \le i \le k$,
*(5)* $\frac{1}{4(H_k+1)} \le a_1 \le \frac{1}{H_k}$, *where* $H_k = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ *denotes the $k$th harmonic number,*
*(6)* $\frac{1}{4i(H_k+1)} \le a_i$, *for all* $1 \le i \le k$.

*Proof* For a fixed $c > 0$, consider the recursively defined sequence $\{b_i(c)\}_{i=0}^{+\infty}$: $b_0 = 0, b_1 = c, b_{i+1} = 2b_i - b_{i-1} - \frac{1}{b_i}$, where we write $b_i \equiv b_i(c)$ to simplify notation. Let $d_i = b_i - b_{i-1}$. Then, $d_1 = c$, and $d_{i+1} = d_i - \frac{1}{b_i}$. Expanding this recurrence, we have:

$$d_{i+1} = c - \left(\frac{1}{b_1} + \cdots \frac{1}{b_i}\right).$$

$$b_{i+1} = c - \left(\frac{1}{b_1} + \cdots \frac{1}{b_i}\right) + b_i = (i+1)c - \left(\frac{i}{b_1} + \frac{i-1}{b_2} + \cdots + \frac{1}{b_i}\right).$$

First, by a simple inductive argument we observe from the above that for sufficiently large values of $c = b_1$, arbitrarily many of the initial elements of sequences $\{b_i(c)\}$ and $\{d_i(c)\}$ are positive. This implies that for any $k$ there exists $c$ that $d_{k+1}(c) > 0$.

Next, fix $i \ge 3$ and suppose that $d_j > 0$, for all $1 \le j \le i$. Then:

$$b_i \le ic.$$

Thus:

$$d_{i+1} \le c - \left(\frac{1}{c} + \cdots + \frac{1}{ic}\right) = c - \frac{H_i}{c}.$$

From the relation $d_{i+1} > 0$, we obtain $H_i < c^2$, so $i < e^{c^2+1}$. This implies that by adjusting $c \in (0, +\infty)$, we can arbitrarily choose the number of positive initial elements of sequence $\{d_i(c)\}$. Observe that $d_i(c)$, for any fixed index $i$, is a continuous function of the parameter $c$. For $c < \ln k - 1$ we have $d_{k+1}(c) < 0$ and by the intermediate value theorem, there must exist a value of $c$ such that $d_{k+1}(c) = 0$, or equivalently, that $b_{k+1} = b_k$. From now on, we use this value of $c$, only. Observe that $d_{k+1} = 0$ implies that $c = \sum_{i=1}^{k} 1/b_i$.

Now, define $a_i \equiv 1/(cb_i)$, for all $0 \le i \le k+1$. Such a sequence $\{a_i\}$ immediately satisfies conditions (1), (2), and (3). Condition (4) is obtained directly by observing that $a_1 = \frac{1}{c^2}$ and applying the replacement $b_i = 1/(ca_i)$ to the defining recursion of $\{b_i\}$.

Condition (5) may be restated as $H_k \le c^2 \le 4(H_k + 1)$. We have already established that the first of these relations holds, since otherwise we would have $d_{k+1} < 0$.

We will first show by induction that $d_i > c - \frac{2H_{i-1}}{c}$ for all $1 \le i \le e^{c^2/4}$. Indeed, the claim holds for $i = 1$. Suppose it holds for all $1 \le j \le i$. Then:

$$b_j = \sum_{l=1}^{j} d_l > cj - \frac{2}{c}\sum_{l=1}^{j} H_{l-1} = cj - \frac{2}{c}(jH_j - j),$$

$$d_{i+1} = c - \sum_{l=1}^{i} \frac{1}{b_l} > c - \sum_{l=1}^{i} \frac{1}{cl - \frac{2}{c}(lH_l - l)}.$$

Since $i < e^{c^2/4}$, then for any $l < i$ we have:

$$H_l < \log l + 1 < \frac{c^2}{4} + 1,$$
$$\frac{2}{c}(lH_l - l) < l\frac{c}{2}.$$

Thus

$$d_{i+1} > c - \sum_{l=1}^{i} \frac{1}{cl - l\frac{c}{2}} = c - \frac{2H_i}{c},$$

and the inductive claim holds. Now if $i < e^{\frac{c^2}{4}-1}$, then:

$$d_i > c - \frac{2H_{i-1}}{c} > c - \frac{2\log i + 2}{c} > c - \frac{c^2}{2c} = \frac{c}{2}.$$

Thus $k > e^{\frac{c^2}{4}-1}$, and we have:

$$c^2 \leq 4(\log k + 1) \leq 4(H_k + 1).$$

Since $b_i \leq ic$ then $a_i \geq 1/(ic^2)$ thus sequence $\{a_i\}$ satisfies condition (6). □

We are now ready to analyze a specific initialization, for which the $k$-agent rotor-router covers the ring particularly slowly. We show here that the worst-case exploration time is $\Theta(\frac{n^2}{\log k})$ for $k < n^{1/11}$. The case of $k > n^{1/11}$ has been considered after publication of the conference version of this paper in [21], where authors showed that the rotor-router explores the ring in time $\Theta\left(\max\left\{n, \frac{n^2}{\log k}\right\}\right)$ for all values of $k$.

**Theorem 1** *In the case when all the agents are initially placed at the same node $v$, a group of $k$ agents explores the ring of size $n$ in time $\Theta(\frac{n^2}{\log k})$ when $k < n^{1/11}$, when all pointers are initialized along the shortest path to $v$.*

*Proof* In the following we will assume that $k \geq 10^6$, since for $k < 10^6$ we have by the slow-down lemma: $C(R[10^6]) \leq C(R[k]) \leq C(R[1])$. It has been shown that $C(R[1]) = \Theta(n^2)$ [27], hence showing $C(R[10^6]) = \Theta(n^2)$ will also imply that $C(R[k]) = \Theta(n^2)$ for any $k < 10^6$.

Since $C(R[k-1]) \geq C(R[k]) \geq C(R[k+1])$, and the cover time is also monotonous with respect to the size of the ring, without affecting asymptotic bounds we can assume that $k$ is even and $n$ is odd. By induction, we can show that the number of agents at node $v$ will be even at all times, and the arrangement of pointers on the ring (except for node $v$) is symmetric with respect to the axis of symmetry passing through $v$. Consequently, the cover time for the $n$-node ring with $k$ agents is asymptotically the same as the cover time of a $(n+1)/2$-node path with $k/2$ agents, starting from an initial placement of all agents on one of the end-points $v$ of the path.

For simplicity consider deployment $R[k]$ of $k$ agents on $n$-node path $P_n$. We now propose a delayed deployment $D$ of $R[k]$ in which, starting from a certain moment in time, the domains of all agents are separate. Let the domains be ordered along the path according to decreasing numbers, i.e., the agent with domain $V_k$ is the one located closest to the starting point $v$, while the agent with domain $V_1$ is the farthest from $v$, i.e., it is the only agent to explore previously unvisited nodes of the path. The goal of the formalization below is to define the delayed deployment so that the ratios of domain sizes satisfy $|V_i| \sim a_i$, for $k \geq i \geq 1$, throughout time, where $\{a_i\}$ is the sequence introduced in Lemma 13.

We will identify the path $P_n$ with the integer interval $[1, n]$ (with $v = 1$), and domains with subsets of this interval. For $k \geq i \geq 1$, let $p_i = \sum_{j=i}^{k} a_i$. For a given value $S$, $n \geq S > 0$, we will call a configuration of agents and pointers on the path a *desirable configuration of length $S$* if it has the following properties:

- The position of the $i$th agent on the path is $v_i = \lfloor p_i S \rfloor$.
- Each agent is at the right endpoint of its domain, i.e., $V_k = [1, v_k]$ and $V_i = [v_{i+1} + 1, v_i]$ for $k - 1 \geq i \geq 1$.
- For all the nodes on the path (including those containing agents), except for node 1, the pointer points to the left (towards node 1).

The evolution of the delayed deployment $D$ is defined in two phases, as follows:

- *Phase* A. Form a desirable configuration with $S_0 = \frac{n}{\sqrt{k}\log k}$. To achieve this, release the agents one-by-one, starting from agent 1 to agent $k$, and perform exactly $(\lfloor p_i S_0 \rfloor - 1)^2$ moves with each agent, so that each agent $i$ occupies position $\lfloor p_i S_0 \rfloor$ and all pointers on the path point to the left.
- *Phase* B. For successive $j = 0, 1, \ldots$, iterate the following procedure, until the path has been covered. Starting from an initial desirable configuration of some length $S_j$, form a new desirable configuration of length $S_{j+1} = S_j + \lceil k^4 a_1 a_k \rceil + 12k$ as follows:

  B1. Starting from the current desirable configuration, release all the agents simultaneously for $\lceil 2k^4 a_k S_j \rceil$ rounds.

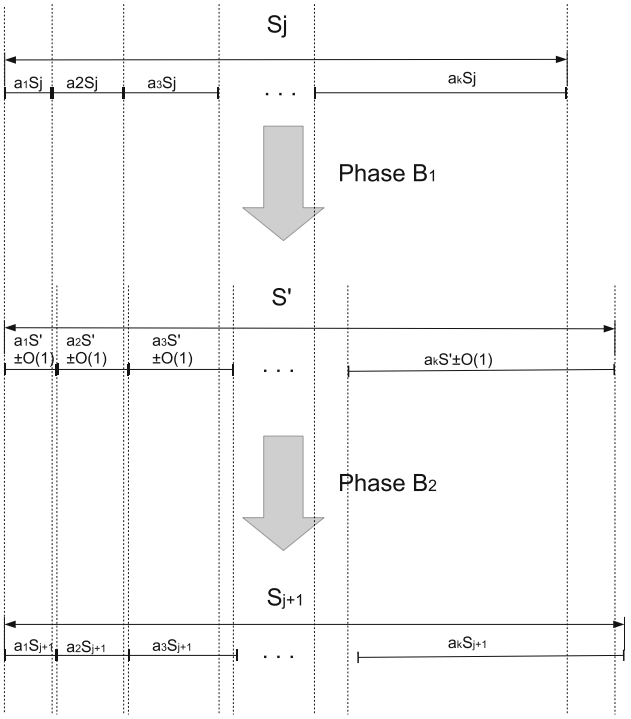


**Fig. 2** An iteration of Phase B of delayed deployment $D$ (Proof of Theorem 1)

B2. Adjust the positions of the agents, so as to reach the desirable configuration of length $S_{j+1}$. To achieve this, release the agents one-by-one, starting from agent 1 to agent $k$, allowing each agent $i$ to move until it has reached position $\lfloor p_i S_{j+1} \rfloor$.

We denote by $T$ the cover time of deployment $D$, by $A$, the total number of rounds of Phase A, by $B_1$, the total number of rounds of Phase B1, and by $B_2$, the total number of rounds of Phase B2. We also remark that during Phase B1 none of the agents is delayed, hence, by Lemma 3 we have:

$$B_1 \leq C(R[k]) \leq T = A + B_1 + B_2.$$

We begin by bounding time $A$. The agents are released sequentially in Phase A. The number of rounds required for each agent to reach its position is less than $\frac{n^2}{k \log k}$. Thus, $A < \frac{n^2}{\log k}$.

We now proceed to Phase B (see Fig. 2 for an illustration). The size of the smallest domain in configuration $S_0$ is:

$$\left\lfloor \frac{n}{\sqrt{k \log k}} a_k \right\rfloor \geq \left\lfloor \frac{n}{\sqrt{k \log k}} \frac{1}{4(H_k+1)k} \right\rfloor \geq \left\lfloor \frac{k^{11}}{k^{3/2} \log k \, (4 \log k + 8)} \right\rfloor \geq k^9,$$

where we used the fact that $n \geq k^{1/11}$ and $k \geq 10^6$. Consider now the $j$th step of the phase, starting from length $S = S_j$, and the change of the configuration within part B1 of this step. The number of rounds used in part B1 of the step is $2a_k S k^4$. Let $|V_i|_j = \lfloor p_i S \rfloor - \lfloor p_{i+1} S \rfloor \geq a_i S - 1$ be the size of the domain of the $i$th agent at the beginning of the $j$th step, and let $|V_i|_j + g_i$ be its size after completion of part B1 of this step. In order to increase the size of its domain, the $i$th agent needs to perform at least $g_i$ traversals of its domain (such that during these traversals the size of this domain is at least $|V_i|_j$), where a traversal is understood as starting and ending at the right endpoint of the domain. These traversals require more than $a_i S g_i$ rounds, whereas the total duration of part B1 of the $j$th step is $\lceil 2a_k S k^4 \rceil$, hence we obtain $g_i < 2k^4$. Since the total size of all domains is non-decreasing in time, it follows that $\sum_{i=1}^{k} g_i \geq 0$, and so:

$$-2k^5 \leq g_i < 2k^4.$$

We now proceed to refine this bound on $g_i$. Initially, the size of the $i$th domain is between $a_i S - 1$ and $a_i S + 1$. Thus, for the $i$th agent, the number of completed traversals $c_i$ of its domain during the considered part B1 is:

$$\frac{2a_k S k^4}{a_i S + 1 + 2k^4} \leq c_i \leq \frac{2a_k S k^4 + 1}{a_i S - 1 - 2k^5}.$$

If the $i$th node performed $c_i$ complete traversals, then it reached each of the boundaries of its domain at least $c_i$ times and one boundary could be reached $c_i + 1$ times. Thus, considering the change in size of domain $g_i$ during the traversals of agents $i$, $i-1$ and $i+1$, we have:

$$2c_i - c_{i-1} - c_{i+1} - 2 \leq g_i \leq 2c_i + 1 - c_{i-1} - c_{i+1}$$

and introducing the bounds on $c_i, c_{i-1}, c_{i+1}$ to the above

$$2a_k S k^4 \left( \frac{2}{a_i S + 1 + 2k^4} - \frac{1}{a_{i-1} S - 1 - 2k^5} - \frac{1}{a_{i+1} S - 1 - 2k^5} \right) - 2 \leq g_i$$

$$g_i \leq \left(2a_k S k^4 + 1\right) \left( \frac{2}{a_i S - 1 - 2k^5} - \frac{1}{a_{i-1} S + 1 + 2k^4} - \frac{1}{a_{i+1} S + 1 + 2k^4} \right) + 1.$$

$$2a_k S k^4 \left( \frac{2}{a_i S} \left( 1 - \frac{2k^4 + 1}{a_i S + 1 + 2k^4} \right) - \frac{1}{a_{i-1} S} \left( 1 + \frac{2k^5 + 1}{a_{i-1} S - 1 - 2k^5} \right) - \frac{1}{a_{i+1} S} \left( 1 + \frac{2k^5 + 1}{a_{i+1} S - 1 - 2k^5} \right) \right) - 2 \leq g_i$$

$$g_i \le 2a_k Sk^4\left(\frac{2}{a_i S}\left(1+\frac{2k^5+1}{a_i S-1-2k^5}\right)\right.$$
$$-\frac{1}{a_{i-1}S}\left(1-\frac{2k^4+1}{a_{i-1}S+1+2k^4}\right)$$
$$\left.-\frac{1}{a_{i+1}S}\left(1-\frac{2k^4+1}{a_{i+1}S+1+2k^4}\right)\right)+2$$

We know that $a_i S \ge k^9$ and $a_k \le a_i$

$$2a_k k^4\left(\frac{2}{a_i}\left(1-\frac{2}{k^5}\right)-\frac{1}{a_{i-1}}\left(1+\frac{2}{k^4}\right)\right.$$
$$\left.-\frac{1}{a_{i+1}}\left(1+\frac{2}{k^4}\right)\right)-3 \le g_i$$
$$g_i \le 2a_k k^4\left(\frac{2}{a_i}\left(1+\frac{2}{k^4}\right)-\frac{1}{a_{i-1}}\left(1-\frac{2}{k^5}\right)\right.$$
$$\left.-\frac{1}{a_{i+1}}\left(1-\frac{2}{k^5}\right)\right)+3$$
$$2a_k k^4\left(\frac{2}{a_i}-\frac{1}{a_{i-1}}-\frac{1}{a_{i+1}}\right)-11-\frac{8}{k} \le g_i \le$$
$$\le 2a_k k^4\left(\frac{2}{a_i}-\frac{1}{a_{i-1}}-\frac{1}{a_{i+1}}\right)+11+\frac{8}{k}$$
$$2a_i a_k k^4 a_1 - 11 - \frac{8}{k} \le g_i \le 2a_i a_k k^4 a_1 + 11 + \frac{8}{k}$$

The above analysis shows that the position of the $i$th agent after Phase B1 is upper-bounded by:

$$\begin{aligned}\lfloor p_i S\rfloor + \sum_{l=i}^{k} g_l &< \lfloor p_i S\rfloor + \sum_{l=i}^{k}(a_l a_1 a_k k^4 + 11 + 8/k)\\ &\le p_i S + p_i a_1 a_k k^4 + 11k + 8\\ &\le \lfloor p_i S_{j+1}\rfloor - k + 9\\ &< \lfloor p_i S_{j+1}\rfloor.\end{aligned}$$

and lower-bounded by:

$$\begin{aligned}\lfloor p_i S\rfloor + \sum_{l=i}^{k} g_l &\ge \lfloor p_i S\rfloor + \sum_{l=i}^{k}(a_l a_1 a_k k^4 - 11 - 8/k)\\ &\ge p_i S + p_i a_1 a_k k^4 a_1 - 11k - 9 \ge\\ &\ge \lfloor p_i S_{j+1}\rfloor - 23k - 9 > \lfloor p_i S_{j+1}\rfloor - 24k.\end{aligned}$$

Now, consider the duration of the Phase B2. Each agent must adjust its position to the right, by a distance of at most $24k$. First, the right-most agent (agent 1) has to perform at most $24k$ traversals of its domain. As a result, the size of the domain of the penultimate agent (agent 2) can decrease by at most $24k$, hence it must perform at most $24k + 24k = 48k$ traversals to reach its position at the end of the step. In general, agent $i$ has to perform at most $24ki \le 24k^2$ traversals of its domain. The size of the $i$th domain during Phase B2 is at most $a_i S + 2k^4 + 48k^2$. Thus, the duration of Phase B2 is bounded by:

$$\sum_{i=1}^{k}\left(a_i S + 2k^4 + 48k^2\right)24k^2 < 24Sk^2 + 48k^7 + 1152k^5.$$

Observe that the duration of part B1 of the step was $2a_k Sk^4 \ge \frac{2}{4k(H_k+1)}Sk^4 > 24Sk^2$ for $k > 10^3$, because $a_k \ge \frac{1}{4k(H_k+1)}$ from Lemma 13. Thus, overall we have that the execution of B1 dominates the complexity of the algorithm, $B_1 \in \Omega(B_2)$ and $B_1 \in \Omega(A)$. It follows that $C(R[k]) = \Theta(B_1)$. Now, in order to bound time $B_1$, observe that the $j$th step of Phase B results in the increase of $S_j$, the number of already covered nodes, by $\Theta(k^4 a_1 a_k)$, which means that Phase B consists of $\Theta\left(\frac{n}{k^4 a_1 a_k}\right)$ steps. Since more than half of these steps are performed for $n/2 < S_j < n$, we obtain a tight bound on the cover time $B_1 \in \Theta(\frac{n^2}{a_1})$. Noting that $a_1 = \Theta(\frac{1}{H_k})$ by Lemma 13, we eventually obtain $B_1 \in \Theta(\frac{n^2}{\log k})$. Thus, $C(R[k]) \in \Theta(\frac{n^2}{\log k})$. □

We now show that the initialization considered above, with all agents starting from one node and all ports pointing to the left, is indeed asymptotically the worst possible. The proof of this theorem proceeds in two steps, first by considering agents starting from one node with an arbitrary placement of pointers on the ring, and then by extending this result to the general case through the application of delayed deployments.

**Lemma 14** *In the case when all the agents are initially placed at the same node v, a group of k agents explores the ring of size n in time $O(\frac{n^2}{\log k})$ when $k < n^{1/11}$, regardless of the initial placement of pointers.*

*Proof* We extend the proof of the upper bound from Theorem 1 to different initializations of pointers. We consider the case of the rotor-router deployment $R[k]$ on the $n$-node path with all agents initially positioned at the left endpoint of the path (but with arbitrary pointer initialization along the path). As in the proof of Theorem 1, we consider a delayed deployment with similarly defined Phases A and B, using the same set of desirable configurations of length $S_j$. Note that in a desirable configuration, all the pointers along the path point to the left for all nodes which have already been visited by an agent at least once. In Phase A, agents are released one-by-one, until the $i$th agent reaches position $\lfloor p_i S_0\rfloor$, after which the agent is stopped (this may happen after a smaller number of steps than in the proof of Theorem 1). In the $j$th step of Phase B, the only difference concerns the definition of part B1, where we add the condition that, upon reaching position $\lfloor p_i S_{j+1}\rfloor$ for the first time, the $i$th agent stops and waits for the other agents to complete part B1 of the step. By induction, one can show that for $i > 1$, agent $i$ will only

stop moving in part B1 after agent $i-1$ has stopped moving, and consequently, it may never happen that a moving agent meets a stationary agent. The analysis of the time spent within parts A, B1 and B2 is performed as before, and we obtain $C(R[k]) = A + B_1 + B_2 = O\left(\frac{n^2}{\log k}\right)$.

The analysis on the ring proceeds by a modification of the argument for a path, treating the ring as two sub-paths connected at the common node 1. In Phase B, the deployments on both sub-paths are synchronized so that the agents $a_k$ of the respective deployments arrive at node 1 simultaneously. If agent $a_k$ of one of the sub-paths, say the left one, arrives before the agent $a_k$ of the right sub-path, then all the agents of the left sub-path are stopped at their current locations until the other agent $a_k$ arrives at node 1. (Note that the two sub-paths do not have to be performing the same step $j$ of Phase B at the same time.) Let $P$ be a path constructed based on the ring by splitting node 1 into two nodes (not connected by an edge). Let $D_l$ and $D_r$ be the two (delayed) deployments of $k/2$ agents exploring a path left-to-right and right-to-left respectively. We can view our transformation as exploration of the path by both deployments from both endpoints. Observe that in our transformation we further delay the deployments $D_l$ and $D_r$ but in each step we make a step in at least one deployment. We know that each of $D_l$ and $D_r$ would explore the path in $O\left(\frac{n^2}{\log k}\right)$ hence the cover time of the deployment on the ring is also $O\left(\frac{n^2}{\log k}\right)$. □

**Theorem 2** *For any initialization of the $k$-agent rotor-router system on the ring, the cover time is $O\left(\frac{n^2}{\log k}\right)$, for $k < n^{1/11}$.*

*Proof* Let $R[k]$ be a deployment of the rotor-router on the ring. Fix a subset $P \subset V$ of $|P| = k^{2/3}$ points on the ring which are evenly spaced, i.e., $G[V \setminus P]$ is a set of disjoint paths of length at most $n/k^{2/3}$. Consider a delayed deployment of $R[k]$, which begins with a Phase in which the agents of $R[k]$ are activated and moved one by one, stopping each agent as soon as it has reached a node from $P$. Since the cover time of a path of length $O(n/k^{2/3})$ for a single agent is $O(n^2/k^{4/3})$, the duration of this Phase is at most $O(n^2/k^{1/3})$. After this initial phase, by the pigeon-hole principle, there must exist a node $v \in P$ which contains $k' \geq k^{1/3}$ agents. We now continue the delayed deployment by releasing $k'' = \min\{k^{1/3}, n^{1/11}\}$ agents which are located at $v$, and permanently stopping (removing) all other agents. By Theorem 1, the path will be covered by the delayed deployment within $O\left(\frac{n^2}{\log k''}\right)$ rounds. By summing the duration of the two phases and using the slow-down lemma, we obtain the claim: $C(R[k]) \in O\left(\frac{n^2}{k^{1/3}} + \frac{n^2}{\log k''}\right) = O\left(\frac{n^2}{\log k}\right)$, for $k < n^{1/11}$. □

### 3.2 Best-case initial placement

We start by proposing the initialization with agents equally spaced along the path as a candidate for (asymptotically) best-case initial placement with $O\left((\frac{n}{k})^2\right)$ cover time. The proof is straightforward in the case if we assume that the adversary initially directs all pointers towards the nearest agent, so as to block it. However, the adversary may apply a different strategy, and there do indeed exist port arrangements which deflect agents from some section of the ring, leading to a larger value of cover time. In our proof we show such actions of the adversary do not affect the asymptotics of the cover time.

**Theorem 3** *Consider an initialization of the rotor-router system on the ring with agents starting on a set of points $P = \{p_1, p_2, \ldots, p_k\}$, such that $G[V \setminus P]$ is a set of paths of length at most $n/k$. Then, the system covers all of the nodes of the ring in time $O\left((\frac{n}{k})^2\right)$, regardless of the initial pointer arrangement.*

*Proof* W.l.o.g, let $1 \leq p_1 < p_2 < \cdots < p_k \leq n$. Given a fixed initial pointer arrangement, let $x \in [1, n]$ be the node which is visited last by the rotor-router. To prove the claim, by the slow-down lemma, it suffices to construct a delayed deployment $D$ of the rotor-router such that point $x$ is visited by some agent within $O\left((\frac{n}{k})^2\right)$ rounds. We define deployment $D$ as follows. Initially, we release all agents simultaneously, so that each agent moves left while the pointer of its current node points to the left, and stops as soon as it encounters a node whose pointer points to the right. Let $q_i$ denote the position of the agent starting from $p_i$ after this phase is complete; we have $p_i - n/k \leq q_i \leq p_i$, hence the duration of this phase is at most $n/k$. We also have $|q_{i+1} - q_i| \leq 2\lceil n/k \rceil$. After this initialization phase, the deployment proceeds in steps of duration $4\lceil n/k \rceil$. The deployment is defined so that at the start of each step, agent $i$ is located at point $q_i$. We describe the deployment through the following procedure, performed simultaneously by each agent $i$. The agent moves (to the right), stopping when it has either reached point $q_{i+1}$, or a node whose pointer points to the left. It then waits until the end of the $2\lceil n/k \rceil$th round of the step to synchronize with other agents, and then returns to node $q_i$, where it waits until the end of the step.

We observe that in each step such that agent $i$ does not reach $q_{i+1}$, it reaches a node on the path $[q_i, q_{i+1}]$ which has not previously been visited by any agent. Suppose that $i$ is such that $q_i < x < q_{i+1}$. It follows that node $x$ will be visited by agent $i$ within $|q_{i+1} - q_i| \leq 2\lceil n/k \rceil$ steps. Since the duration of each step is $4\lceil n/k \rceil$, the second phase of the delayed deployment takes at most $8\lceil n/k \rceil^2$ round. Overall, point $x$ is covered within $O\left((\frac{n}{k})^2\right)$ rounds from the start of the process, and the claim follows. □

To prove that the equally-spaced initialization is asymptotically the best possible, we provide a general case lower-bound of $\Omega\left((\frac{n}{k})^2\right)$ on cover time for all initializations. To do this, we introduce an auxiliary notion of a *remote vertex* for an initialization of the rotor-router. Intuitively a vertex $v$ is remote if for any $i$, the segment of $x_i$ vertices $[v - x_i/2, v + x_i/2]$ contains at most $10 \cdot x_i k/n$ initial positions of the agents (where values $x_i$ are appropriately chosen). Such vertices are shown to always exist (in fact, to be in the majority in the vertex set) and take a long time to cover, regardless of the initial placement of agents.

**Definition 2** For any placement of the $k$ agents let $S = \{s_1, s_2, \ldots, s_k\}$ be the $k$ not necessarily distinct starting vertices. We will consider the subset of *remote vertices* of the cycle, defined as all nodes $v$ which satisfy the following two constraints:

1. For all $1 \leq r \leq k$, $\left|\left[v, v + r\frac{n}{10k}\right] \cap S\right| \leq r$.
2. For all $1 \leq r \leq k$, $\left|\left[v, v - r\frac{n}{10k}\right] \cap S\right| \leq r$.

The following lemma concerning the relation between remote vertices and the starting positions of the agents, and proves useful in the analysis of the $k$-agent rotor-router, as well as the $k$-agent random walk.

**Lemma 15** *If $k = \omega(1)$, then for any initial placement $S = \{s_1, s_2, \ldots, s_k\}$ of the $k$ agents, there are at least $0.8n - o(n)$ remote vertices.*

*Proof* Let $V_1$ and $V_2$ be the sets of vertices which satisfy constraints 1 and 2 above, respectively. We first show that $|V_1| \geq 0.9n - o(n)$. Consider an algorithm which starts from vertex 0 and scans the cycle in the increasing order of vertex numbers, as follows:

1. $v \leftarrow 0$
2. $B \leftarrow \emptyset$
3. While $(v < n - (n/10k))$, repeat:
   If $v \notin V_1$, then:
   (a) Let $r$ be the smallest positive integer such that $|[v, v + r(n/k)) \cap S| > r$.
   (b) $B \leftarrow B \cup [v, v + r(n/10k))]$
   (c) $v \leftarrow v + r(n/k)$
   else $v \leftarrow v + 1$.

By the construction of set $B$, each new interval of the form $[v, v + r(n/10k))$, of length $r(n/10k)$ which is added to it, contains more than $r$ elements of set $S$. Consequently: $|B \cap S| > 10k|B|/n$, so $|B| < 0.1n|S|/k = 0.1n$. On the other hand, we observe that for $v < n-(n/10k)$, $v \notin B \implies v \in V_1$, and so $|V_1| \geq n - o(n) - |B| \geq 0.9n - o(n)$. By a similar argument, we show that $|V_2| \geq 0.9n - o(n)$. From here, we obtain the sought bound on the number of remote vertices: $|V_1 \cap V_2| \geq 0.8n - o(n)$. □

**Theorem 4** *If $n \geq 440k^2$, then for any set of initial locations of $k$ agents, there exists an initial arrangement of pointers on the ring such that the cover time of the rotor-router system is $\Omega\left((\frac{n}{k})^2\right)$.*

*Proof* If $k = O(1)$, then in any initial placement there is a subpath of $cn$ vertices without any agent for some constant $c > 0$. If we initialize the pointers on this path negatively then exploration of this subpath will be no faster than the exploration of a $cn$-node ring with all agents starting at the same node which is $\Omega(n^2)$ by Theorem 1. From now on we will assume that $k = \omega(1)$. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the $k$ not necessarily distinct starting vertices. Let $r_i$ be the number of vertices initially between $s_i$ and $s_{i+1}$, and $r_k$ be the number of vertices between $s_k$ and $s_1$. Obviously $\sum_i r_i \geq n - k$. Thus $\sum_{\{i: r_i \geq \frac{n-k}{2k}\}} r_i \geq \frac{n-k}{2}$. If we take two middle quarters from each interval of length at least $\frac{n-k}{2k}$ then totally we will obtain at least $\frac{n-k}{4}$ nodes. Thus at least $n/4 - o(n)$ nodes are at distance at least $\frac{n-k}{8k}$ to the closest agent. If $n \geq 9k$ then $\frac{n-k}{8k} \geq \frac{n}{9k}$. Thus by Lemma 15 there are at least $0.05n - o(n)$ remote nodes at distance at least $\frac{n}{9k}$ to the closest starting point of an agent. For sufficiently large $n$ such node will exist. We will call this node $v$. Now we will use Lemma 1 and construct a delayed deployment $D1$. We will block all but one or two agents to ensure that each agent will have a domain of size at least $\frac{n}{20k}$ and at least $\frac{n}{10k}$ nodes will not be explored. We initiate all pointers negatively—in each node the pointer points away from the closest agent. We will describe the procedure in one direction. In the other direction procedure will be the same. Firstly we release the closet agent at the left of $v$ until it reaches the node at distance $\frac{n}{20k}$ from $v$. Then we block the agent. Since the closest node to $v$ is at distance at least $\frac{n}{9k}$ then after this procedure in interval $[v + \frac{n}{20k}, v + \frac{n}{10k}]$ there will be only one agent. Then we take the next closest agent at the left of $v$ and release it until it reaches node $v + \frac{n}{10k}$. Again since $v$ is a remote node there will be only one agent in interval $[v + \frac{n}{10k}, v + \frac{n}{5k}]$. Then for $i$th closest agent at the left of $v$ for $i \geq 2$ we release it until it reaches node $v + (i-1)\frac{n}{10k}$. It is possible, that the agent will go to the other side of the ring. Then we block it at the node $v + \frac{n}{2}$ and continue procedure. We do the same procedure to the left and right from $v$. We end up with some agents at node $v + \frac{n}{2}$. We release them one-by-one. Assume that such agent $a$ went to the left from $v$. We block him, when he is at distance $\frac{n}{10k}$ from the last agent placed to the left of $v$. Now each agent has a domain of size at least $\frac{n}{20k}$. Now we release all agents simultaneously. By Lemma 8, size of any domain will not drop below $\Omega(\frac{n}{20k})$. Assumptions of Lemma 8 are satisfied, because $\frac{n}{20k} \geq 10k$ and the pointers are initialized negatively. We also have a group of $\frac{n}{20k}$ not explored nodes.

Since this group will be explored by agents having domains of sizes at least $\Omega\left(\frac{n}{20k}\right)$ it will take at least $\Omega\left(\frac{n^2}{400k^2}\right)$ time steps. Number of steps in $D1$ when all agents are released simultaneously is $\Omega\left(\frac{n^2}{k^2}\right)$, thus by Lemma 1 the cover time of not delayed $k$ agents in the rotor-router model in this case will also be $\Omega\left(\frac{n^2}{k^2}\right)$. □

### 3.3 Comparison with the random walk

The question of the cover time of random walks starting from a worst-case initial placement has already been resolved in the literature. On the one hand, it is known that the speed-up of cover time for a $k$-agent random walk with respect to the single agent case is $\Omega(\log k)$ for any graph whose cover time is asymptotically equal to the maximum hitting time [4], regardless of the initial placement of agents. Since this is clearly the case for the ring [2], we have that the cover time of the $k$-agent random walk is $O(n^2/\log(k))$. On the other hand, the adversary may choose to place all agents at one node of the ring. Such an all-on-one initialization has a cover time of precisely $\Theta(n^2/\log(k))$ [4]. Thus, the cover time for $k$ random walks on the ring with worst-case initialization is $\Theta(n^2/\log(k))$.

To establish an upper bound for the best-case scenario, we consider $k$ random walks with initial positions given with equal spacing, i.e., with offsets equal to $0, \frac{n}{k}, 2\frac{n}{k}, \ldots, (k-1)\frac{n}{k}$ relative to some node. (For simplicity, we assume here that $k$ divides $n$.) The following lemma implies that in this case the cover time is $O((n/k)^2)$.

**Lemma 16** *Let $\alpha \geq 20$, $k \geq 2$ and let $t := \alpha^2 \cdot (n/k)^2 \cdot \log^2(k)$. Then, with probability at least $1-k^{1-\alpha/20}$, $k$ random walks starting from initial positions with equal spacing cover all the vertices of the ring within $t$ steps.*

*Proof* Recall that $t = \alpha^2 \cdot (n/k)^2 \cdot \log^2(k)$. Since the maximum hitting time of a single random walk on a path with $\frac{1}{5}\sqrt{t}+1$ nodes is at most $\frac{1}{25}t$ (cf. [22]), we conclude from Markov's inequality that a single random walks on the ring with $n$ vertices visits a vertex which is at least $\frac{1}{5}\cdot\sqrt{t}$ to the right of its starting vertex within $t$ steps with probability at least $1/4$. Note that for any vertex $u \in V = \{0,\ldots,n-1\}$, there are at least $x-1$ random walks with distance between $(n/k)$ and at most $x\cdot(n/k)$ to $u$. Putting $x = \frac{1}{10}\cdot\sqrt{t}/(n/k)$ we obtain the following upper bound for the event that $u$ will *not* be covered:

$$\left(1-\frac{1}{4}\right)^{\frac{1}{10}\cdot\sqrt{t}/(n/k)-1} = \left(1-\frac{1}{4}\right)^{\frac{\alpha}{10}\cdot\log(k)-1} \leq k^{-\alpha/20}.$$

Now note that if for any vertex $u$ of the set $S := \{0, n/k, 2(n/k), \ldots, (k-1)(n/k)\}$ there is a random walk that is initially placed to the left of $u$ with distance at most $\frac{1}{10}\cdot\sqrt{t}/(n/k)$ and which traverses at least $\frac{1}{5}\cdot\sqrt{t}$ steps to the right within the first $t$ steps, then all vertices of the ring are covered after $t$ steps. Hence by taking the union bound over the set $S$ we conclude that all vertices of the ring are covered with probability at least $1-k\cdot k^{-\alpha/20} = 1-k^{1-\alpha/20}$. □

We now prove a corresponding lower bound on the cover time in the best-case scenario, showing that the position with equal spacing is asymptotically the best possible. We first prove an auxiliary result which relies on the notion of remote vertices introduced in the previous subsection.

**Lemma 17** *Let $t = 10^{-4}\cdot(n/k)^2\cdot\log^2(k)$, $k=\omega(1)$, and let $u$ be any remote vertex at distance at least $\frac{n}{10k}$ from the starting points of all random walks. Then, with probability at least $k^{-1/2}$, $u$ is not covered after $t$ steps by any of the $k$ random walks.*

*Proof* Consider first a random walk with the number of steps equal to $(n/k)/10 \leq d \leq 4\cdot\sqrt{t}$ to $u$ (recall that $t = 10^{-4}\cdot(n/k)^2\cdot\log^2 k$). We are interested in the probability that the random walk reaches a point with distance at least $4\cdot\sqrt{t}$ to $u$ before visiting $u$. Using the Gambler's ruin problem, this probability is equal to $\frac{d}{4\cdot\sqrt{t}}$. Once the random walk has distance $4\cdot\sqrt{t}$ to $u$, the probability that it does not visit $u$ within $t$ steps is at least $1/2$ (this follows by using a standard Chernoff bound). Combining these insights, we obtain that a random walk with distance $d \leq 4\cdot\sqrt{t}$ does not visit the vertex $u$ within $t$ steps with probability at least $\frac{d}{4\cdot\sqrt{t}}\cdot\frac{1}{2}$. Consider now *all* random walks with distance less than $4\cdot\sqrt{t}$. The number of these random walks is $4\cdot\sqrt{t}/(n/k) = (1/25)\log k$. The probability that none of these random walks visits $u$ is at least

$$\begin{aligned}
&\prod_{j=0}^{(1/25)\log(k)-1} \frac{\frac{n}{10k}+j\cdot\frac{n}{10k}}{4\cdot\sqrt{t}}\cdot\frac{1}{2}\\
&\geq 4^{-\frac{\log k}{25}} \prod_{j=1}^{(1/25)\log(k)} \frac{j\cdot\frac{n}{10k}}{\sqrt{t}}\\
&= 4^{-\frac{\log k}{25}}\cdot\prod_{j=1}^{(1/25)\log k}\frac{j}{(1/10)\log k}\\
&= 10^{-\frac{\log k}{25}}\cdot\prod_{j=1}^{(1/25)\log k}\frac{j}{(1/25)\log k}\\
&\geq 10^{-\frac{\log k}{25}}\cdot\frac{((1/25)\log k)!}{((1/25)\log k)^{(1/25)\log k}}\\
&\geq 10^{-\frac{\log k}{25}}\cdot e^{-\frac{\log k}{25}},
\end{aligned}$$

where the last line follows from Stirling's approximation, i.e., for any sufficiently large integer $m\in\mathbb{N}$, $m! \geq (m/e)^m$.

For a random walk with distance $d = c\cdot\sqrt{t}$, $c\geq 4$ to $u$, the probability to visit $u$ is at most $e^{-c/2}$ by a Chernoff

bound. Hence the probability that $u$ is visited by none of the random walks with distance at least $4 \cdot \sqrt{t}$ is lower bounded by

$$\begin{aligned}
&\prod_{j=(1/25)\log k}^{k} \left(1 - e^{-\frac{j \cdot \frac{n}{k}}{\sqrt{t}} \cdot \frac{1}{2}}\right) \\
&= \prod_{j=(1/25)\log(k)}^{k} \left(1 - e^{-\frac{j}{\frac{1}{100}\log k} \cdot \frac{1}{2}}\right) \\
&= \prod_{j=(1/25)\log(k)}^{k} \left(1 - e^{-\frac{50j}{\log k}}\right)^{e^{\frac{50j}{\log k}} \cdot e^{-\frac{50j}{\log k}}} \\
&\geq 4^{-\sum_{j=(1/25)\log k}^{\infty} e^{-\frac{50j}{\log k}}} \\
&\geq 4^{-\frac{\log k}{25} \cdot \sum_{j=1}^{\infty} e^{-2j}} \geq 4^{-\frac{\log k}{25}},
\end{aligned}$$

where the third line used the fact that $(1-x)^{1/x} \geq \frac{1}{4}$ for $x \leq 1/2$. Hence none of the $k$ random walks will visit $u$ with probability at least

$$10^{-(1/25)\log(k)} \cdot e^{-(1/25)\log(k)} \cdot 4^{-\frac{\log(k)}{25}} \geq k^{-1/2}.$$

□

The lower bound on cover time is completed when we prove the existence of a remote vertex satisfying the conditions of Lemma 17. We do this taking into account Lemma 15. □

**Lemma 18** *For arbitrary starting positions of $k$ random walks, we need at least $\Omega((n/k)^2 \log^2 k)$ steps to visit all $n$ vertices with probability at least $1/2$.*

*Proof* If $k = O(1)$ we can find a path of $n/k = \Omega(n)$ vertcies not containing any starting position of a walk. Using similar arguments as in Lemma 17 we get that time $\Omega(n^2)$ is needed in this case. Now assume that $k = \omega(1)$. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the $k$ not necessarily distinct starting vertices. Fix $t = 10^{-4} \cdot (n/k)^2 \cdot \log^2(k)$. We define intervals $I_i, 0 \leq i < k/\log^2 k$, of the form $I_i = [(i-1)(n/k)\log^2 k, i(n/k)\log^2 k)$. The length of the union of all intervals with even indices is $I = \bigcup_{0 \leq j < k/2\log^2 k} I_{2j}$, and $|I| = 0.5n - o(n)$.

If $F$ is the set of remote vertices, then by Lemma 15, $|F| \geq 0.8n - o(n)$. Let $H$ be the set of all nodes at distance at least $(n/k)/10$ to node from $S$; we have $|H| \geq 0.8n$. Thus $|I \cap F \cap H| \geq 0.1n - o(n)$ and $|I \cap F \cap H| \geq 0.09n$ for sufficiently large $n$. Since each interval $I_i$ is of length $(n/k)\log^2 k$, at least $0.09k/\log^2 k$ intervals with even indices must contain a remote vertex satisfying assumptions of lemma 17. We pick one such vertex from each interval. In this way, we obtain set $S$ of $0.09k/\log^2 k$ vertices, at pairwise distances of at least $(n/k)\log^2 k$ from each other.

We denote by $Y$ the event that none of random walks reached a distance more than $d = 40 \cdot \sqrt{t}\log k$ to its origin. We note that $\mathbf{Pr}[Y] \geq 1 - k^{-40}$. We also denote by $X$ the event, that every vertex in $S$ is explored in time $t$ by $k$ random walks. Note, that $\mathbf{Pr}[X|Y] \leq \left(1 - k^{-1/2}\right)^{\frac{k}{\frac{1}{10}\log^2(k)}}$ because if event $Y$ happened, then each vertex $s \in S$ remains uncovered with probability at least $k^{-1/2}$. These events are independent for different vertices in $S$ because $d = 0.4(n/k)\log^2 k$ thus if event $Y$ happens then none of the walks can visit more than one vertex from $S$. Hence

$$\begin{aligned}
\mathbf{Pr}[X] &\leq \mathbf{Pr}[Y]\,\mathbf{Pr}[X|Y] + 1 - \mathbf{Pr}[Y] \\
&\leq \left(1 - k^{-1/2}\right)^{\frac{k}{\frac{9}{100}\log^2(k)}} + k^{-40} \\
&\leq 1/2
\end{aligned}$$

The last inequality holds for $k > 1$. □

Now, the characterization of the cover time of $k$ random walks in the best-case scenario follows directly by Lemmas 16 and 18.

**Theorem 5** *The cover time of $k$ random walks on the ring for best-case initial placement is $\Theta((n/k)^2 \log^2 k)$.* □

## 4 Return time of the rotor-router on the ring

The considerations of the rotor-router in the previous section concerned the time required to cover all nodes in the initialization phase. As a deterministic system with a finite number of states, the rotor-router eventually reaches its limit behavior, cycling through a finite number of configurations. In this section, we characterize this limit behavior of the rotor-router using the concept of *return time*, i.e. the maximum over $v \in V$ of the length of the longest time interval during which $v$ is not visited by any agent of the rotor-router system in its limit behavior. We show that this performance parameter of the rotor-router on the ring achieves the best possible value of $\Theta(\frac{n}{k})$, regardless of the initial placement of the agents.

**Theorem 6** *If $k \in O\left(n^{1/6}\right)$ then after a sufficiently large number of time steps, the $k$-agent rotor-router system will visit every node of the $n$ vertex ring once every $\Theta(\frac{n}{k})$ time steps.*

*Proof* In this proof we will make use of delayed deployments of agents. When analyzing delayed deployments, we apply the same definition of a domain as in Sect. 2.2. We will construct a delayed deployment which will ensure that

each agent has domain of size at least $20k$ and then we will release all the agents. After all the agents are released, all the lemmas from Sect. 2.2 hold unchanged.

We will denote the undelayed deployment by $R[k]$ and a specific delayed deployment by $D$. The considered deployment $D$ consists of two phases. In the first phase we release all the agents until all the nodes are visited. In the second phase, we selectively release (and delay) agents until each agent has a domain of size at least $20k + 2$; details of the construction are provided later. In the final third phase, we release all the agents. By Lemma 12, the size of every domain will eventually converge to $O\left(\frac{n}{k}\right)$. Thus, in deployment $D$ every node will eventually be visited once every $O\left(\frac{n}{k}\right)$ time steps.

Now we describe the construction for the first phase of deployment $D$ to achieve domains of size at least $20k+2$, so that not all agents are active at the same time in at most $O\left(\frac{n}{k}\right)$ rounds. We proceed as follows. First, we release all agents until all nodes of the ring have been covered. Next, we release agents one by one, progressing the agent until it has reached a point located a distance of at least $40k+4$ from the nearest agent. Since the longest sub-path consisting of agents, which do not have a gap of length at least $2(40k + 6) + 1$ between them, is $80k^2 + 13k$, and the moving agent is equivalent to a single-agent rotor-router system, the agent will reach the endpoint of such a sub-path (and complete its movements) within $(80k^2 + 13k)^2$ steps. In total, the moves of all agents in this stage of the construction require $O(k^5)$ rounds. In the next stage, we deploy the agents one by one, so that each agent is moved until it is located at a distance of precisely $20k + 2$ from its location at the beginning of this stage. By a similar analysis, the duration of this stage is $O(k^3)$. Note that during this stage no two agents meet or pass each other on an edge, and so each agent is adjacent to a path of length $20k+2$ with ports arranged towards its current location. Hence, we have achieved $|V_a(t)| \geq 20k + 2$ within a total of $O(k^5)$ steps, which is $O(n/k)$ for $k \in O\left(n^{1/6}\right)$.

It remains to show that for the undelayed deployment $R[k]$ the return time is also $O(n/k)$. Let $\theta$ be the total number of rounds during which not all agents are active in $D$. The construction of $D$ ensures that $\theta \in O\left(\frac{n}{k}\right)$. Now, let $t$ be a time step such that after $t$ every node is visited at least once every $c\frac{n}{k}$ time steps by some agent following deployment $D$, for some constant $c > 0$. Take any $t^* > t$. We have that in $D$, every node is visited at least once in the time interval $\left[t^*, t^* + c\frac{n}{k}\right]$. By Lemma 3 we have that for any $v$, $n_v^{R[k]}(t^* - \theta) \leq n_v^D(t^*)$, and $n_v^{R[k]}\left(t^* + c\frac{n}{k}\right) \geq n_v^D\left(t^* + c\frac{n}{k}\right) > n_v^D(t^*)$. Thus, for any time $t^*$, some agent following $R[k]$ visits $v$ within the time interval $[t^* - \theta, t^* + c\frac{n}{k}]$, which contains $t^*$ and is of duration $O\left(\frac{n}{k}\right)$. It follows directly that the refresh time of $R[k]$ is $O\left(\frac{n}{k}\right)$. □

No strong analogue of the above theorem holds for a system with $k$ random walks, which is not deterministic by nature and with positive probability a vertex can remain unvisited for an arbitrary number of steps. A natural property which can be bounded for the random walk is the expected time between two successive visits to a node, which is precisely equal to $n/k$ on the ring (since the stationary distribution of each of the $k$ walks is uniform with probability $1/n$ on each node). However, the random variable which describes the expected time between successive visits to a node has high variance.

## 5 Conclusions

We have shown that the multi-agent rotor-router and the parallel random walk have similar speed-up characteristics w.r.t. the number of deployed agents, at least in terms of cover time and return time on the ring. It is interesting to note that the worst-case speed-up on the ring is $\Theta(\log k)$ for both the $k$-agent random walk and the $k$-agent rotor-router, even though this speed up has a different explanation in both cases. For the random walk, it is a consequence of the properties of probability distributions of independent Markovian processes, while for the rotor-router, it results directly from the interactions between different agents and the pointers in the graph. On the other hand, the best-case speed-up of $\Theta(\frac{k^2}{\log^2 k})$ for the random walk takes into account a poly-logarithmic factor which results from the probabilistic nature of the walk, whereas the speed-up for the rotor-router is simply $\Theta(k^2)$.

Our work may also be seen as a step in the direction of understanding and characterizing the behavior of the multi-agent rotor-router in graphs different from the ring. Some of the techniques developed in this paper, in particular analysis based on delayed deployments, are also applicable in the general case.

**Acknowledgements** A. Kosowski and R. Klasing thank Leszek Gąsieniec and Tomasz Radzik for inspiring discussions. The authors would like to thank the anonymous reviewers for their comments.



## Appendix

### Proof of Lemma 8

*Proof* We will perform the proof by induction, showing the following two implications. We will first show that if condition (1) holds for all steps $\tau = 1, 2, \ldots, t$ then (2)

holds for step $t+1$ and if condition (2) holds for all steps $\tau = 1, 2, \ldots, t$ then (1) holds for step $t$. We first show the second implication. Assume that in step $t$, agent $a$ moves the border between $a$ and $b$. Take $\tau = t - 2|V_b'(t)| - 8$, assume that $\tau \geq t_0$, and consider the interval of time $[\tau+1, t]$ of length $2|V_b'(t)| + 8$. We first want to show that within this interval each domain border is moved at most once. Assume, to the contrary, that some agent $c$ moved one of its borders at least twice. Then, by the properties of agent motion within domains (Propositions 1 and 2), the agent visited its whole lazy domain 4 times. But by the inductive assumption, the lazy domain of $c$ is of size at least $\mu - 5k + 5$ within the considered time interval. Thus, $4\mu - 20k + 20 \leq 2|V_b'(t)| + 8 \leq 2\mu + 2$, a contradiction, since $\mu \geq 10k$. Thus, within this time interval, the size of the lazy domain of $b$ was at most $|V_b'(t)| + 2$.

Observe that if $\tau < t_0$, then using the same argument within the time interval $[t_0, t]$, each domain border on the ring was moved at most once. Since $|V_b'(t)| \leq \mu - 3$, one of the borders of the domain of $b$ must have been moved at least twice since time $t_0$, a contradiction.

This implies that agent $b$ visited the whole of its domain within the considered time interval, including the domain border with $a$. On the other hand, the size of the lazy domain of $a$ was at least $|V_a'(t)| - 2 \geq |V_b'(t)| + 6$, thus $a$ could visit the border only once within the interval $[\tau+1, t]$. Now, by Proposition 1, agent $a$ cannot move the border in step $t$.

To show the second implication, consider the following one-player *token game*. We have $k$ stacks, each initially containing $\eta$ tokens. A move in the game consist in moving a token from one stack to another. A move is legal if the stack to which the token is moved contains at most 8 tokens more than the one from which the token is taken. The token game is similar to classical chip-firing games on graphs [5]. We make use of the following key property of the token game. □

**Claim** After any number of legal moves of the token game, each stack contains at least $\eta - 5k + 5$ *tokens.*

*Proof (of claim)* Let $x_1^{(t)}, x_2^{(t)}, \ldots, x_k^{(t)}$ denote the sizes of the stacks after $t$ moves, ordered so that the sequence $x^{(t)}$ is non-increasing: $x_1^{(t)} \geq x_2^{(t)} \geq \cdots \geq x_k^{(t)}$. Let $X^{(t)}$ be the multiset of all stack heights $x_i^{(t)}$, $1 \leq i \leq k$. Let $y_i^{(t)} = \sum_{j=k-(i-1)}^{k} x_j^{(t)}$, $1 \leq i \leq k$, denote the sum of the $i$ largest elements of $X^{(t)}$. We will show that the following invariant holds after any number of legal moves: for all $i \in \{1, 2, \ldots, k\}$, we have $y_i^{(t)} \leq \eta i + 5ki - 5i^2$. The proof proceeds by induction over time. Observe that for $t = 0$ we have $y_i^{(0)} = \eta i$, and the inequalities hold for all $i$, since $k \geq i$. Now, assume by contradiction that there exists a sequence of moves such that, after $t+1$ moves, the invariant is not true for the first time in the game, and this violation occurs for some index $j \in \{1, 2, \ldots, k\}$. Observe that $j < k$, because $y_k^{(t)}$ represents the total sum of all stack heights, which does not change during the game. Consider the state of the game after $t$ steps. We have $y_j^{(t)} = \eta j + 5kj - 5j^2$, since the inductive claim is violated in step $t+1$ by assumption, and the sum of the $j$ largest values can increase by at most one when moving a single token. We also have $y_i^{(t)} \leq \eta i + 5ki - 5i^2$, for all $1 \leq i \leq k$ by the inductive assumption. Let $W_1$ be the multiset of the $j$ largest values of $X^{(t)}$ and let $W_2$ be the multiset of remaining $k - j$ values of $X^{(t)}$.

We have:

$$\eta(j+1) + 5k(j+1) - 5(j+1)^2 \geq y_{j+1}^{(t)} = y_j^{(t)} + x_{j+1}^{(t)} = \eta j + 5kj - 5j^2 + x_{j+1}^{(t)},$$

and we get $x_{j+1}^{(t)} \leq \eta + 5k - 10j - 5$. Next, if $j = 1$ we have $x_1^{(t)} = y_1^{(t)} = \eta + 5k - 5 = \eta + 5k - 10j + 5$. If $j > 1$ we have:

$$\eta j + 5kj - 5j^2 = y_j^{(t)} = y_{j-1}^{(t)} + x_j^{(t)} \leq \leq \eta(j-1) + 5k(j-1) - 5(j-1)^2 + x_j^{(t)},$$

and so in either case, $x_j^{(t)} \geq \eta + 5k - 10j + 5$. We thus obtain $x_j^{(t)} \geq x_{j+1}^{(t)} + 10$. Consequently, the difference between the smallest value in $W_1$ and the largest in $W_2$ is at least 10. Thus, any move which takes a token from $W_2$ and moves it to $W_1$ is not legal. Again, using the fact that $x_j^{(t)} \geq x_{j+1}^{(t)} + 10$, we observe that any move of a token performed in step $t+1$, which must be a move within $W_1$ or $W_2$ or from $W_1$ to $W_2$, cannot increase $y_j^{(t)}$. We have thus obtained a contradiction; hence, the invariant holds for all $t$ by induction. Now, since $y_{k-1}^{(t)} \leq \eta(k-1) + 5(k-1)k - 5(k-1)^2 = \eta(k-1) + 5k - 5$ and the total number of tokens is $\eta k$, we have directly that the smallest stack contains always at least $\eta - 5k + 5$ tokens. This completes the proof of the claim.

Now, consider values $w_a(t) = \min\{|V_a'(t)|, \mu - 3\}$ being the sizes of the stacks of tokens in the token game. By the inductive assumption (1) we know that a vertex from the lazy domain of agent $a$ can be captured by agent $b$ only if $|V_a'(t)| > \mu - 3$ or $|V_a'(t)| + 8 > |V_b'(t)|$. Thus, if $w_a(t) + 8 \leq w_b(t)$, the move from lazy domain of $a$ to lazy domain of $b$ is illegal. Thus we can view the set of lazy domain sizes over time as a special case of an instantiation of the token game. □

**Proof of Lemma 9**

*Proof* Assume that $a$ moves its border with $b$ in step $t$ and let $t^*$ be the time of the previous visit of $a$ to this border. By Lemma 7 we have a lower and an upper bound on $t - t^*$; however, the upper bound depends on the size of the lazy domain $V_b'$ at time $t^*$. In the interval $[t^*, t]$ agent $c$ could have captured some nodes of the lazy domain of $b$. In the

following we want to bound the number of nodes that could be captured by $c$ in the interval $[t^*, t]$.

We denote $i_c = \min_{s\in[t^*,t]}\{|V_c'(s)|\}$, representing the minimum size of lazy domain of agent $c$ in time interval $[t^*, t]$. We will consider two cases. First assume that $i_c > |V_b'(t^*)|/3$, thus $t - t^* \le 2|V_b'(t^*)| + 3 \le 6i_c + 3$ and $c$ can make at most 3 complete traversals of its domain in each direction. Hence $c$ visits the border with $b$ at most 4 times. By Proposition 1 $b$ can lose at most 2 nodes to $c$ in time interval $[t^*, t]$, thus $|V_b'(t^*)| - 2 \le |V_b'(t)|$. We have:

$$\begin{aligned} t - t^* &\le 2|V_b'(t^*)| + 3 \le 2|V_b'(t)| \\ &\quad + 7 \le 2|V_a'(t)| - 11 < t - t^*, \end{aligned}$$

which leads to a contradiction.

Now consider the case when $i_c \le |V_b'(t^*)|/3$. This means that agent $c$ during interval $[t^*, t]$ increased the size of its lazy domain from at most $|V_b'(t^*)|/3$ to at least $|V_b'(t)|/2$. To increase the size of the lazy domain from $i_c$ to $i_c + 1$, agent $c$ needs to make at least one full traversal of its domain in each direction and must visit the border at least twice, thus at least $2i_c + 2$ steps are needed. By iterating the argument, in order to increase the lazy domain size from $i_c$ to $i_c + \delta$, the agent needs at least $2\delta(i_c + \delta + 1)$ steps. Thus, to increase the lazy domain size from $i_c$ to $|V_c'(t)|$, at least $2\left\lfloor \frac{|V_c'(t)|-i_c}{2} \right\rfloor \left(i_c + \left\lfloor \frac{|V_c'(t)|-i_c}{2} \right\rfloor + 1\right)$ steps are needed. Thus, we have $t - t^* \ge 2\left\lfloor \frac{|V_c'(t)|-i_c}{2} \right\rfloor \left(i_c + \left\lfloor \frac{|V_c'(t)|-i_c}{2} \right\rfloor + 1\right)$. On the other hand, $t - t^* \le 2|V_b'(t^*)| + 3$, and we have:

$$\begin{aligned} 2|V_b'(t^*)| + 3 &\ge t - t^* \\ &\ge 2\left\lfloor \frac{|V_c'(t)| - i_c}{2} \right\rfloor \\ &\quad \times \left(i_c + \left\lfloor \frac{|V_c'(t)| - i_c}{2} \right\rfloor + 1\right) \\ &\ge 2\left(\frac{|V_c'(t)| - i_c}{2} - 1\right)\left(i_c + \frac{|V_c'(t)| - i_c}{2}\right) \\ &= |V_c'(t)|(|V_c'(t)| - 2)/2 - i_c^2/2 - i_c \\ &\ge 16/33|V_c'(t)|^2 - i_c^2/2 - i_c \\ &\ge 4/33|V_b'(t)|^2 - |V_b'(t^*)|^2/18 - |V_b'(t^*)|/3 \end{aligned}$$

In the above we used the inequality $|V_c'(t)| \ge 15k + 5 \ge 66$ (thus $2 \le 1/33|V_c'(t)|$) which follows from Lemma 7. Thus, we have

$$|V_b'(t^*)|^2/6 + 7|V_b'(t^*)| + 15 \ge 4/11|V_b'(t)|^2 \tag{4}$$

Since $a$ has performed one complete traversal in the time interval $[t^*, t]$, all nodes that $b$ lost during this interval must have been taken by agent $c$. Thus, agent $c$ had to perform at least $2i_c(|V_b'(t^*)| - |V_b'(t)|)$ steps. By Lemma 8 we have:

$$\begin{aligned} 2|V_b'(t^*)| + 2 &\ge t - t^* \ge 2i_c(|V_b'(t^*)| - |V_b'(t)|) \\ &\ge 132(|V_b'(t^*)| - |V_b'(t)|), \end{aligned}$$

and consequently:

$$132|V_b'(t)| + 2 \ge 134|V_b'(t^*)|. \tag{5}$$

By combining inequalities 4, 5, and the inequality $|V_b'(t^*)| \ge 66$, we obtain a contradiction. □

## References


1. Akbari, H., Berenbrink, P.: Parallel rotor walks on finite graphs and applications in discrete load balancing. In: Blelloch, G.E., Vöcking, B. (eds.) 25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'13, Montreal, QC, Canada, July 23–25, 2013. ACM, pp. 186–195 (2013)
2. Aldous, D., Fill, J. A.: Reversible Markov Chains and Random Walks on Graphs (1995) http://stat-www.berkeley.edu/users/aldous/RWG/book.html
3. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovasz, L., Rackoff, C.: Random walks, universal traversal sequences, and the complexity of maze problems. In: Proceedings of the 20th Annual Symposium on Foundations of Computer Science, FOCS'79, Washington, DC, USA, 1979. IEEE Computer Society, pp. 218–223 (1979)
4. Alon, N., Avin, C., Koucký, M., Kozma, G., Lotker, Z., Tuttle, M.R.: Many random walks are faster than one. Comb. Probab. Comput. **20**(4), 481–502 (2011)
5. Anderson, R., Lovasz, L., Shor, P., Spencer, J., Tardos, E., Winograd, S.: Disks, balls, and walls: analysis of a combinatorial game. Am. Math. Mon. **96**(6), 481–493 (1989)
6. Bampas, E., Gasieniec, L., Hanusse, N., Ilcinkas, D., Klasing, R., Kosowski, A.: Euler tour lock-in problem in the rotor-router model. In: DISC, LNCS, vol. 5805, pp. 423–435 (2009)
7. Bampas, E., Gasieniec, L., Klasing, R., Kosowski, A., Radzik, T.: Robustness of the rotor-router mechanism. In: OPODIS, LNCS, vol. 5923, pp. 345–358 (2009)
8. Berenbrink, P., Klasing, R., Kosowski, A., Mallmann-Trenn, F., Uznanski, P.: Improved analysis of deterministic load-balancing schemes. In: Georgiou, C., Spirakis, P.G. (eds.) Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21–23, 2015. ACM, pp. 301–310 (2015)
9. Bhatt, S.N., Even, S., Greenberg, D.S., Tayar, R.: Traversing directed eulerian mazes. J. Graph Algorithms Appl. **6**(2), 157–173 (2002)
10. Broder, A.Z., Raghavan, P., Taylor, R.W., Karlin, A.R., Karlin, A.R., Upfal, E., Upfal, E.: Trading space for time in undirected s-t connectivity. In: Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, pp. 543–549 (1991)
11. Cooper, C., Ilcinkas, D., Klasing, R., Kosowski, A.: Derandomizing random walks in undirected graphs using locally fair exploration strategies. Dist. Comp. **24**(2), 91–99 (2011)
12. Cooper, J.N., Spencer, J.: Simulating a random walk with constant error. Comb. Probab. Comput. **15**(6), 815–822 (2006)
13. Doerr, B., Friedrich, T.: Deterministic random walks on the two-dimensional grid. Comb. Probab. Comput. **18**(1–2), 123–144 (2009)

14. Efremenko, K., Reingold, O.: How well do random walks parallelize? In: APPROX-RANDOM, LNCS, vol. 5687, pp. 476–489 (2009)
15. Elsässer, R., Sauerwald, T.: Tight bounds for the cover time of multiple random walks. Theor. Comput. Sci. **412**(24), 2623–2641 (2011)
16. Feige, U.: A spectrum of timeSpace trade-offs for undirected s-t connectivity. J. Comput. Syst. Sci. **54**(2), 305–316 (1997)
17. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. Theor. Comput. Sci. **345**(2–3), 331–344 (2005)
18. Friedrich, T., Sauerwald, T.: The cover time of deterministic random walks. Electr. J. Comb. **17**(1), R167 (2010)
19. Gasieniec, L., Radzik, T.: Memory efficient anonymous graph exploration. In: WG, LNCS, vol. 5344, pp. 14–29 (2008)
20. Klasing, R., Kosowski, A., Pajak, D., Sauerwald, T.: The multi-agent rotor-router on the ring: a deterministic alternative to parallel random walks. In: PODC. ACM, pp. 365–374 (2013)
21. Kosowski, A., Pajak, D.: Does adding more agents make a difference? A case study of cover time for the rotor-router. In: Automata, Languages, and Programming—41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8–11, 2014, Proceedings, Part II, Lecture Notes in Computer Science, vol. 8573. Springer, pp. 544–555 (2014)
22. Lovász, L., Winkler, P.: A note on the last new vertex visited by a random walk. J. Graph Theory **17**(5), 593–596 (1993)
23. Priezzhev, V., Dhar, D., Dhar, A., Krishnamurthy, S.: Eulerian walkers as a model of self-organized criticality. Phys. Rev. Lett. **77**(25), 5079–5082 (1996)
24. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 17 (2008)
25. Sauerwald, T.: Expansion and the cover time of parallel random walks. In: PODC. ACM, pp. 315–324 (2010)
26. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. **15**, 918–933 (1999)
27. Yanovski, V., Wagner, I.A., Bruckstein, A.M.: A distributed ant algorithm for efficiently patrolling a network. Algorithmica **37**(3), 165–186 (2003)