Lock-in Problem for Parallel Rotor-router Walks

Jérémie Chalopin, Shantanu Das, Pawel Gawrychowski, Adrian Kosowski, Arnaud Labourel, Przemyslaw Uznanski

Lock-in Problem for Parallel Rotor-router Walks

Jérémie Chalopin[1], Shantanu Das[1], Paweł Gawrychowski[2], Adrian Kosowski[3], Arnaud Labourel[1], and Przemysław Uznański[1]

[1]LIF, CNRS and Aix-Marseille University, France
[2]Max-Planck-Institut für Informatik, Saarbrücken, Germany
[3]Inria – LIAFA – Paris Diderot University, France

Abstract

The *rotor-router* model, also called the *Propp machine*, was introduced as a deterministic alternative to the random walk. In this model, a group of identical tokens are initially placed at nodes of the graph. Each node maintains a cyclic ordering of the outgoing arcs, and during consecutive turns the tokens are propagated along arcs chosen according to this ordering in round-robin fashion. The behavior of the model is fully deterministic. Yanovski *et al.*(2003) proved that a single rotor-router walk on any graph with m edges and diameter D stabilizes to a traversal of an Eulerian circuit on the set of all $2m$ directed arcs on the edge set of the graph, and that such periodic behaviour of the system is achieved after an initial transient phase of at most $2mD$ steps.

The case of multiple parallel rotor-routers was studied experimentally, leading Yanovski *et al.* to the conjecture that a system of $k > 1$ parallel walks also stabilizes with a period of length at most $2m$ steps. In this work we disprove this conjecture, showing that the period of parallel rotor-router walks can in fact, be exponential in the size of graph. On the positive side, we provide a characterization of the periodic behavior of parallel router walks, in terms of a structural property of stable states called a *subcycle decomposition*. This property provides us the tools to efficiently detect whether a given system configuration corresponds to the transient or to the limit behaviour of the system. Moreover, we provide a polynomial upper bound of $\mathcal{O}(m^4D^2 + mD\log k)$ on the number of steps it takes for the system to stabilize. Thus, we are able to predict any future behavior of the system using an algorithm that takes polynomial time and space. In addition, we show that there exists a separation between the stabilization time of the single-walk and multiple-walk rotor-router systems, and that for some graphs the latter can be asymptotically larger even for the case of $k = 2$ walks.

1 Introduction

The exploration of unknown graphs is a well-studied problem that arises in various different contexts in computer science, *e.g.*, finding a path between two vertices, checking connectivity, or in the broader context of searching for information. In particular, searching for information on the internet graph of webpages is often performed by software agents (crawlers) that move from node to node on the graph. The agent starts at a node of the graph and at each step moves to one of the adjacent nodes until it has explored all the nodes of the graph. In some cases, we may require the agent to periodically visit all nodes (*e.g.* for monitoring a network). There is a motivation for designing mechanisms for exploration of large graphs using simple light weight agents, having only constant memory (independent of the graph size). We call such an agent a *token*. One useful technique is the *random walk* where at each step the token chooses uniformly at random one of the incident arcs at the current node and traverses it. The random walk performs well in expectation and it is practically useful and easy to implement since the token does not need to have any memory. One way to speed up the exploration process is to employ multiple tokens in parallel. Parallel random walks have been analyzed by Alon *et al.* [3], Efremenko and Reingold [11], and Elsasser and Sauerwald [12], achieving a speedup between $\Theta(\log k)$ and $\Theta(k)$ for random walks by k parallel tokens.

However, randomization has its own cost and in this paper we are interested in efficient *deterministic* methods for graph exploration. *The rotor-router mechanism* was introduced as a deterministic alternative to a random walk, and was studied in a variety of contexts such as load balancing problems [6, 8, 10], graph exploration [1, 9, 14, 16, 18], and stabilization of distributed processes [4, 7, 19, 23]. The rotor-router mechanism is represented by an undirected anonymous graph $G = (V, E)$. The nodes are unlabelled, however each node v keeps endpoints of edges at v (called *ports*) arranged in a permanent *cyclical order*. Furthermore, each node is equipped with a *pointer* that indicates the current exit port to be taken whenever a token is propagated from the node. Initially, a set of identical tokens is released on a vertex of the graph. At discrete, synchronous steps, the tokens are propagated according to the deterministic rules of rotor-router mechanism. One such rule is the round robin rule where after sending each token, the pointer is advanced to the next exit port in the fixed cyclic ordering. It is known that using such a simple rule is sufficient to allow for exploration of any connected graph with any arbitrary ordering of the outgoing arcs at each vertex. Since the system is fully deterministic and has finite number of possible states, it will exhibit a regular behavior, *i.e.*, after a finite number of stabilization steps, a sequence of states will be repeated forever. A natural question arise of structural behavior of system in repeated states, alongside of questions of length of stabilization time and periodic behavior.

The behavior of a single rotor router walk has been completely analyzed in previous works (see below). In this paper, we study the parallel rotor router walks and provide a set of tools to characterize the behavior of $k > 1$ parallel rotor router walks in arbitrary graphs.

Related work The rotor-router model has been previously studied as the *Propp machine* [4], as an *Edge Ant Walk algorithm* [21, 23], and in the context of traversing a maze and marking edges with pebbles, *e.g.* in [7]. Exploration time of rotor-router systems have been investigated by Wagner *et al.* [21] who showed that starting from an arbitrary initial configuration[1] the token explores any graph of n nodes and m edges, within $\mathcal{O}(nm)$ steps. Later, Bhatt *et al.* [7] showed that after at most $\mathcal{O}(nm)$ steps, the token continues to move periodically along an Eulerian cycle of the (directed symmetric version) of the graph. Yanovski *et al.* [23] and Bampas *et al.* [4] studied the stabilization time and showed that the token starts circulating in the Eulerian cycle within $\Theta(mD)$ steps, in the worst case, for a graph of diameter D. Studies of the rotor router system for specific classes of graphs were performed in [15]. While all these studies were restricted to static graphs, Bampas *et al.* [5] considered the time required for the rotor-router to stabilize to a new Eulerian cycle after an edge is added or removed from the graph.

Studies of the parallel (*i.e.*, multiple token) rotor-router was performed by Yanovski *et al.* [23] and Klasing *et al.* [17], and its speedup was considered for both worst-case and best-case scenarios. In [9], Dereniowski *et al.* establish bounds on the minimum and maximum possible cover time for a worst-case initialization of a k-rotor-router system in a graph G with m edges and diameter D, as $\Omega(mD/k)$ and $\mathcal{O}(mD/\log k)$ respectively. In [18], Kosowski and Pająk provided a more detailed analysis of the speedup for specific classes

[1] A configuration is defined by the cyclic order of outgoing arcs, the initial pointers at the nodes and the current location of the token.

of graphs, providing tight bounds of cover-time speed-up for all values of k for degree-restricted expanders, random graphs, and constant-dimensional tori. For hypercubes, they resolve the question precisely, except for values of k much larger than n.

Parallel rotor-router walks have also been studied in the context of balancing the workload in a network. In this case, each token is a single task to be performed by one of the processors in a network of computers. Most studies have been restricted to specific graph classes. For example, Cooper and Spencer [8] studied parallel rotor walks in d-dimensional grid graphs and showed a constant bound on the discrepancy 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. The distribution of tokens for a rotor-router system on the 2-dimensional grid, has been analyzed in more details by Doerr and Friedrich [10]. Akbari and Berenbrink [2] proved an upper bound of $\mathcal{O}(\log^{3/2} n)$ on the discrepancy for hypercubes, and for tori of constant dimensions, they showed that the discrepancy is bounded by a constant.

Our results The main contribution of the paper is a characterization of the structural behavior of tokens when the parallel rotor router system enters a stable state, *i.e.*, a state that would be repeated in the future. We show that in this situation, tokens can be divided into indivisible groups that patrol the arcs of the graph, and two groups never traverse the same arc. We name such behavior *subcycle decomposition*, and exact properties are described in Section 3. There are several consequences of this characterization.

We start with providing a counterexample, that admits periods of exponential length with respect to the size of the graph. This counterexample disproves the hypothesis of Yanovski *et al.* [23], that the period length of any rotor-router system is short (at most $2m$). It is worth noticing that the long period in such examples comes from being composed from many small parts each of which exhibits a small (but different) period length. Such structural decomposition lies at heart of the idea of subcycle decomposition.

Second, we show that it is possible to determine efficiently whether the system already stabilized (*i.e.*, reached a configuration that would repeat itself) or not. This detection is based on the analysis of the properties of stable states, that is how the tokens arriving at a node are distributed into groups leaving on different outgoing arcs. The main point of this analysis is the observation that the cumulated size of groups visiting a vertex v (over the time period $[t..t + \Delta t]$) is similar to the cumulated size of groups leaving vertex v (over time $[t + 1..t + \Delta t + 1]$).

Third, by defining an appropriate potential of a system and showing its monotonicity, we can give a polynomial bound on a number of steps necessary for a system with an arbitrary initialization to reach a periodic configuration. We provide an upper bound of $\mathcal{O}(m^4D^2 + mD \log k)$, together with examples of graphs with initial configuration having just 2 tokens that require $\Omega(m^2 \log n)$ steps. This analysis is presented in Section 4.

Finally, the Section 6 is dedicated to showing how the previous results can be applied in a constructive way with regard to efficient simulation of a rotor-router system. We show how the properties of subcycle decomposition can be applied to provide a way to preprocess any starting configuration in a way that makes it possible to answer queries of certain type in a polynomial time. This shows that a structural characterization of the rotor-router system is not only important as a theoretical tool for understanding the limit behavior of the system, but it also as a practical tool for solving certain problems related to the rotor-router system.

As a complementary result, we show how, based on the results from Yanovski *et al.* [23] and Bampas *et al.* [4], concerning behavior of single token under rotor-router rules, we can efficiently compute the Eulerian traversal cycle on which the token would be locked-in. A naive simulation would take $\mathcal{O}(mD)$ time, but by using the structural properties of a single token walk together with application of efficient data structures we show how to preprocess the input graph in time $\mathcal{O}(n + m)$ such that we can answer queries about token position at any given time T, in $\mathcal{O}(\log \log m)$ time per query.

2 Model and preliminaries

Let $G = (V, E)$ be an undirected connected graph with n nodes, m edges and *diameter* D. Let k be the number of tokens. The digraph $\vec{G} = (V, \vec{E})$ is the directed version of G created by replacing every edge (u, v) with two directed arcs $\vec{uv}$ and $\vec{vu}$. We will refer to the undirected links in graph G as *edges* and to

the directed links in the graph $\vec{G}$ as *arcs*. Given a vertex v, we will denote its set of incoming arcs by $\text{in}(v)$ and outgoing arcs by $\text{out}(v)$. Each vertex v of G is equipped with a fixed ordering of all its outgoing arcs $\rho_v = (e_1, e_2, \ldots, e_{\deg(v)})$.

2.1 Token distribution models

In [20] authors introduced the notion of *round fair* algorithms. Specifically, we will call an algorithm *strictly fair* if, in every step, the number of tokens that are sent out over any two edges incident to a node differs by at most one. Round fair algorithms will be useful as a tool of analysis presented in Section 4.

This condition can be strengthened into algorithms which are *cumulatively fair*. We will call an algorithm *cumulatively fair* if for every interval of consecutive time steps, the total number of tokens sent out by a node differs by at most a small constant for any two adjacent edges. Cumulative fair algorithms are an equivalent formulation to the rotor-router model.

2.2 The rotor-router model

We consider the rotor-router model on the system defined by $(\vec{G}, (\rho_v)_{v\in V})$.
State: A *state* at the current timestep t is a tuple

$$\mathcal{H}_t = ((pointer_v)_{v\in V}, (tokens_v)_{v\in V}),$$

where $pointer_v$ is an arc outgoing from node v, which is referred to as *the current port pointer at node* v, and $tokens_v$ is the number of tokens at any given node. For an arc $(\vec{vu})$, let $next(\vec{vu})$ denote the arc next to arc $(\vec{vu})$ in the cyclic order ρ_v. During each step, each node v distributes in a round-robin fashion all of its tokens, using the following algorithm:
While there is a token at node v, do

1. Send token to $pointer_v$,
2. Set $pointer_v = next(pointer_v)$.

2.3 Preliminaries

Multisets We denote multisets using $\{\{\}\}$ notation, for example $\{\{0,0,1\}\} \neq \{\{0,1\}\}$.

Integer ranges We define

$$[a..b] \stackrel{\text{def}}{=} \{a, a+1, \ldots, b\}$$
$$[a..b) \stackrel{\text{def}}{=} \{a, a+1, \ldots, b-1\}$$

Token distribution We will use calligraphic large letters (*e.g.* $\mathcal{L} : \vec{E} \cup V \to \mathbb{Z}$) to denote *token distributions*. We will use specifically $\mathcal{L}_t$ to denote token distribution associated with state $\mathcal{H}_t$. (It is important to note, that it is possible for two states to satisfy $\forall_e \mathcal{L}_t(e) = \mathcal{L}_{t'}(e)$ and yet $\mathcal{H}_t \neq \mathcal{H}_{t'}$, as we also require that pointers be in the same positions in identical states.)

Definition 2.1. *We define the* cumulated token distribution *for a vertex v (arc e) between timesteps $t_1 \leq t_2$ as:*

$$\mathcal{C}_{t_1}^{t_2} \stackrel{\text{def}}{=} \sum_{t=t_1}^{t_2-1} \mathcal{L}_t,$$

in particular: $\mathcal{C}_{t_1}^{t_2}(v) \stackrel{\text{def}}{=} \sum_{t=t_1}^{t_2-1} \mathcal{L}_t(v)$, $\mathcal{C}_{t_1}^{t_2}(e) \stackrel{\text{def}}{=} \sum_{t=t_1}^{t_2-1} \mathcal{L}_t(e)$.

Definition 2.2. *For a given state $\mathcal{H}_t$, we say that it is* stable, *if the system will return to this state in the future (thus, iff there exists $t' > t$ such that $\mathcal{H}_{t'} = \mathcal{H}_t$).*

Definition 2.3. The stabilization time *of state $\mathcal{H}_0$, denoted t_s, is number of steps after it reaches stable state.*

Definition 2.4. *We call* the periodicity time *of state $\mathcal{H}_0$ the smallest $t_p > 0$ such that $\mathcal{H}_{t_s} = \mathcal{H}_{t_s+t_p}$.*

3 Lock-in structure

We begin with observation, that knowledge of first $t_s + t_p$ states, that is $\mathcal{H}_0, \ldots, \mathcal{H}_{t_s+t_p-1}$, gives us full knowledge of any future state for arbitrarily large time $t \geq t_s$:

$$\mathcal{H}_t = \mathcal{H}_{t_s+((t-t_s) \bmod t_p)}$$

If we want to efficiently predict future evolution of any rotor-router state, it would be useful to put a polynomial bound on t_p and t_s (with respect to n, m and $\log k$). If $k = 1$, due to results from Yanovski *et al.* [23], we know that $t_p = 2m$ and $t_s = O(mD)$. For arbitrary k, Yanovski *et al.* [23] conjectured that $t_p \leq 2m$ for any graph G regardless of the initial state. However, the following negative result disproves their conjecture and shows that the periodicity cannot be polynomially bounded for parallel rotor-routers.

Theorem 3.1. *There exists a family of graphs and initial state with $k = 2m$ tokens having the periodicity $t_p = 2^{\Omega(\sqrt{n})}$.*

Proof. We will construct such a family of graphs $\mathcal{G}_r$ for any sufficiently large integer r, and an appropriate initial configuration of tokens. First consider a balloon graph G_x consisting of a cycle of $x > 3$ vertices $\{v_0, v_1, \ldots v_{x-1}\}$ and an additional vertex v_x (called the *base vertex*) that is joined by an edge to vertex v_{x-1} of the cycle (see Figure 1(a)). Let the initial token distribution at vertices $(v_0, \ldots, v_x)$ be $(1, 2, 2, \ldots 2, 4, 1)$. Further let the exit pointers at vertices $v_1, \ldots v_{x-1}$ be oriented in the clockwise direction, while at the vertex v_0 the exit pointer is oriented in the counter-clockwise direction. At the base vertex v_x there is only one outgoing arc and so, the exit pointer at v_x will always point towards this arc.

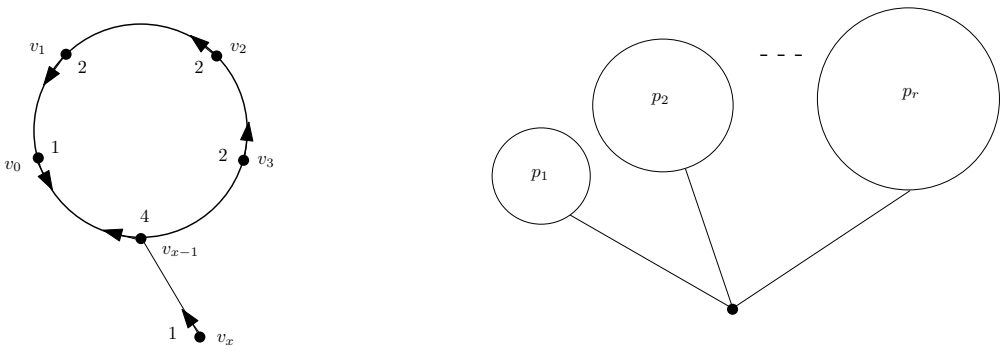
Observe that for a vertex of out-degree two, the exit pointer remains unchanged if two tokens exit this vertex in the current round, while the exit pointer is rotated if an odd number of tokens exit in the current round.

We will now analyze the movement of tokens along the arcs of the graph in each round. Figure 2 shows an example for a balloon graph G_x, where $x = 5$. During the first round, the number of tokens moving on the arcs (v_0, v_1), (v_1, v_2), $\ldots (v_{x-1}, v_0)$ of the cycle (in the clockwise direction) is given by the sequence S_0=$(0, 1, 1, \ldots, 1, 2)$. During the same round, the number of tokens moving on the arcs in the counter-clockwise direction on the cycle is given by $(1, 1, \ldots 1)$. On the branch edge (v_{x-1}, v_x) there is exactly one token moving in each direction (See Figure 2(a)).

During the second round, the number of tokens moving on the arcs (v_0, v_1), (v_1, v_2), $\ldots (v_{x-1}, v_0)$ of the cycle (in the clockwise direction) is given by the sequence S_1=$(1, 1, \ldots, 1, 2, 0)$ which is a cyclic rotation of the sequence S_0. The number of tokens moving on the arcs in the counter-clockwise direction on the cycle is still given by $(1, 1, \ldots 1)$. Again, the branch edge (v_{x-1}, v_x) has exactly one token moving in each direction (See Figure 2(b)).

Continuing with the above analysis, it is easy to see that in subsequent rounds, the number of tokens moving on the arcs of the cycle (in the clockwise direction) is given by cyclic rotations of S_0, *i.e.*, by the sequences $(1, \ldots, 1, 2, 0, 1)$, $(1, \ldots, 1, 2, 0, 1, 1)$, $(1, \ldots, 1, 2, 0, 1, 1, 1)$ and so on (See Figure 2(c-e)). The number of tokens moving along the cycle in the counterclockwise direction is always one token per arc of the cycle. On the branch edge (v_0, v_x) there is exactly one token moving in each direction in each round. Since the length of the sequence S_0 is $|S_0| = x$, after every x steps the configuration of tokens moving on the arcs of the cycle is the same. In other words, the periodicity of this rotor-router system is x. Notice that the graph G_x has $x + 1$ vertices and $2(x + 1)$ arcs, and there are exactly $2(x + 1)$ tokens in the system.

We will now construct the family of graphs $\mathcal{G}_r$. For any given r, let $p_1, p_2, \ldots p_r$ be the first r prime numbers starting from $p_1 = 3$. We take r balloon graphs of sizes $(1 + p_1), (1 + p_2), \ldots, (1 + p_r)$ respectively and join them by merging all the base vertices into one vertex (see Figure 1(b)). In each balloon graph we place the tokens as before, such that the merged base vertex now contains r tokens. During each step, r tokens will exit the base vertex through the r outgoing arcs and r other tokens will enter the base vertex through the r incoming arcs. Thus, irrespective of the initial state of the exit pointer at the base vertex, the system will behave in the same manner. The behavior of the system in the distinct balloons would be independent of each other and for each balloon of size $(1 + p_i)$ the configuration of the balloon would repeat itself in exactly p_i steps as before. Thus, the global state of the system would repeat in $\text{lcm}(p_1, \ldots p_r) = \Pi_{i=1}^{r} p_i$ steps. Note that the size of the graph, $\mathcal{G}_r$, is given by $n = 1 + \sum_{i=1}^{r} p_i = \Theta(r^2 \log r)$. Thus the periodicity t_p of the system can be computed as $t_p = \Pi_{i=1}^{r} p_i = 2^{\Omega(\sqrt{n})}$. □



(a) (b)

Figure 1: (a) The balloon graph and the initial token distribution. (b) The family of graphs $\mathcal{G}_r$ consisting of r balloons.

However, we proceed with a series of equivalent characterization of sufficient and necessary conditions for a state to be stable. Eventually it will show us that even though a stable state can exhibit exponentially long periodicity time, the underlying graph G can be partitioned into parts, such that each part separately exhibits small (linear) periodicity.

Observation 3.2. *Since no tokens are left in a node at any given timestep (every token is pushed onto some outgoing arc),*

$$\sum_{e \in \text{out}(v)} \mathcal{L}_t(e) = \mathcal{L}_t(v); \sum_{e \in \text{in}(v)} \mathcal{L}_t(e) = \mathcal{L}_{t+1}(v)$$

□

Combining Definition 2.1 with Observation 3.2, we obtain:

Observation 3.3.

$$\sum_{e \in \text{out}(v)} \mathcal{C}_{t_1}^{t_2}(e) = \mathcal{C}_{t_1}^{t_2}(v); \sum_{e \in \text{in}(v)} \mathcal{C}_{t_1}^{t_2}(e) = \mathcal{C}_{t_1+1}^{t_2+1}(v)$$

The next observation follows from the fact that rotor-router distributes tokens in a round-robin fashion among all outgoing arcs of a vertex:

Observation 3.4.

$$\forall_{e_1, e_2 \in \text{out}(v)} |\mathcal{C}_{t_1}^{t_2}(e_1) - \mathcal{C}_{t_1}^{t_2}(e_2)| \leq 1$$

Definition 3.5. *Given a token distribution over edges $\mathcal{A}$, we define its potential as:*

$$\Phi(\mathcal{A}) \stackrel{\text{def}}{=} \sum_{e \in \vec{E}} (\mathcal{A}(e))^2$$

We also introduce a shorthand notation for the i-th potential of a given rotor-router state as:

$$\Phi_i(\mathcal{L}_t) \stackrel{\text{def}}{=} \Phi(\mathcal{C}_t^{t+i}) = \sum_{e \in \vec{E}} \left(\mathcal{C}_t^{t+i}(e)\right)^2$$

Note that $\Phi_1 \equiv \Phi$.

We provide following lemma coming from the folklore, as it will be very useful in analysis of rotor-router token distribution.

Lemma 3.6. *Over all partitions of integer S into d integers, the partition into $\{\{\lfloor \frac{S}{d} \rfloor, \ldots, \lfloor \frac{S}{d} \rfloor, \lceil \frac{S}{d} \rceil, \ldots, \lceil \frac{S}{d} \rceil\}\}$ uniquely minimizes the value of sum of squares of elements.*

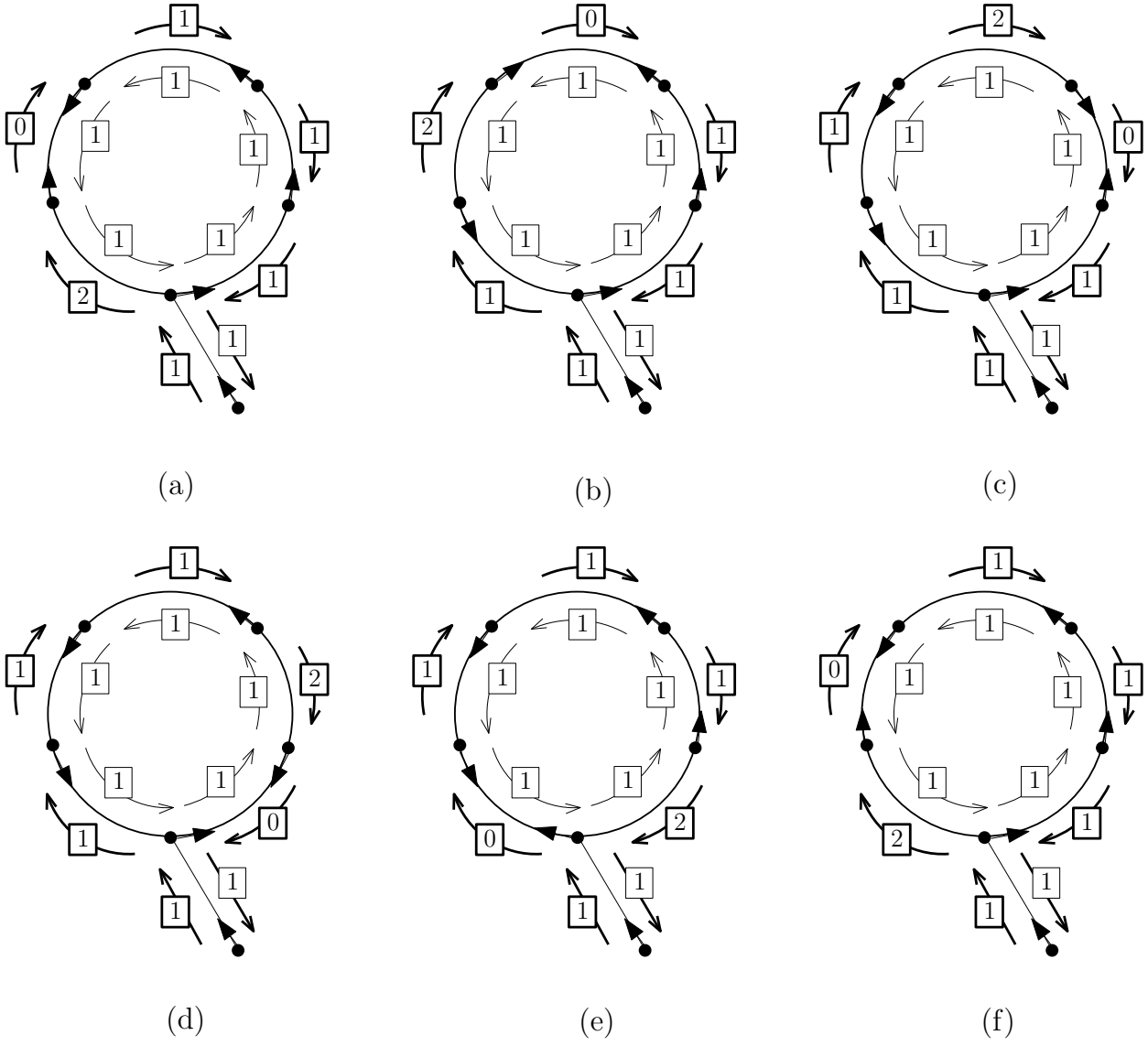


Figure 2: The number of tokens circulating on the arcs of a balloon graph G_5 in each round.

Proof. Let us assume on the contrary, that there exist other minimal partition $A = \{\{a_1, \ldots, a_d\}\}$. As the partition into $\lfloor \frac{S}{d} \rfloor$ and $\lceil \frac{S}{d} \rceil$ is the only one with discrepancy at most 1, there exists pair of elements of A, that is w.l.o.g. a_1 and a_2 such that: $a_1 \geq a_2 + 2$. Thus,

$$a_1^2 + a_2^2 > a_1^2 + a_2^2 - 2(a_1 - a_2 - 1) = (a_1 - 1)^2 + (a_2 + 1)^2,$$

giving that $\{\{a_1 - 1, a_2 + 1, a_3, \ldots, a_d\}\}$ has smaller sum of squares, which contradicts the assumption. □

Lemma 3.7. *For arbitrary i and t, the i-th potential is nonincreasing: $\Phi_i(\mathcal{L}_{t+1}) \leq \Phi_i(\mathcal{L}_t)$.*

Proof. To prove the lemma we have to observe how the round-robin property of the rotor-router acts locally on the groups of tokens (cumulated over the time interval $[t, t+i)$). From Observation 3.3 we know, that:

$$\sum_{e \in \mathrm{out}(v)} \mathcal{C}_{t+1}^{t+i+1}(e) = \mathcal{C}_{t+1}^{t+i+1}(v) = \sum_{e \in \mathrm{in}(v)} \mathcal{C}_t^{t+i}(e)$$

However, from Observation 3.4 and Lemma 3.6 we get that the multiset of values over outgoing arcs *minimizes* the sum of squares. Thus:

$$\sum_{e\in\text{out}(v)} (\mathcal{C}_{t+1}^{t+i+1}(e))^2 \leq \sum_{e\in\text{in}(v)} (\mathcal{C}_t^{t+i}(e))^2,$$

which leads to

$$\Phi_i(\mathcal{H}_{t+1}) = \sum_v \sum_{e\in\text{out}(v)} (\mathcal{C}_{t+1}^{t+i+1}(e))^2 \leq \sum_v \sum_{e\in\text{in}(v)} (\mathcal{C}_t^{t+i}(e))^2 = \Phi_i(\mathcal{H}_t)$$

□

Definition 3.8. *We say that a state $\mathcal{H}_T$ admits a* Δt-step subcycle decomposition*, if in every vertex v we can define a one-to-one mapping between incoming and outgoing arcs of v $M_v : \text{in}(v) \to \text{out}(v)$, such that:*

$$\forall_{e\in\text{in}(v)} \forall_{t\in[T..T+\Delta t)} \mathcal{L}_t(e) = \mathcal{L}_{t+1}(M_v(e)) \tag{1}$$

The subcycle decomposition has the following equivalent interpretation. We partition $\vec{E} = \vec{E_1} \cup \ldots \cup \vec{E_c}$, such that each of $\vec{E_i}$ induces a strongly-connected subgraph of G, and for each of $\vec{E_i}$ there exists an Eulerian cycle covering it such that each group of tokens traversing arcs of $\vec{E_i}$ follow this particular Eulerian cycle during time steps $T, T+1, \ldots, T+\Delta t-1$.

Observe that the mapping M in Definition 3.8 does not need to be necessarily unique. We will call any such a mapping M *a valid mapping with respect to* Δt *subcycle decomposition* if (1) holds for it.

The following lemma gives characterization of subcycle decomposition.

Lemma 3.9. *Following statemets are equivalent:*

(i) $\mathcal{H}_T$ *admits a* Δt*-step subcycle decomposition*

(ii) $\forall_v \forall_{[t..t')\subseteq[T..T+\Delta t)} \{\{\mathcal{C}_t^{t'}(e)\}\}_{e\in\text{in}(v)} = \{\{\mathcal{C}_{t+1}^{t'+1}(e)\}\}_{e\in\text{out}(v)}$ *(multisets of cumulated loads are preserved locally)*

(iii) $\forall_{[t..t')\subseteq[T..T+\Delta t)} [\mathcal{C}_t^{t'}(e)]_{e\in\vec{E}} = \{\{\mathcal{C}_{t+1}^{t'+1}(e)\}\}_{e\in\vec{E}}$ *(multisets of cumulated loads are preserved globally)*

(iv) $\forall_{0\leq i\leq\Delta t} \Phi_i(\mathcal{L}_T) = \Phi_i(\mathcal{L}_{T+1}) = \ldots = \Phi_i(\mathcal{L}_{T+\Delta t-i+1})$ *(potential is constant)*

(v) $\forall_v \forall_{e_1,e_2\in\text{in}(v)} \forall_{[t..t')\subseteq[T..T+\Delta t)} |\mathcal{C}_t^{t'}(e_1) - \mathcal{C}_t^{t'}(e_2)| \leq 1$ *(incoming discrepancy)*

Proof. (i) ⇒ (ii) A straightforward consequence of the Δt-step subcycle decomposition is existence of local bijective mapping $M_v : \text{in}(v) \to \text{out}(v)$ that:

$$\forall_{[t..t')\subseteq[T..T+\Delta t)} \forall_{e\in\text{in}(v)} \mathcal{C}_t^{t'}(e) = \mathcal{C}_{t+1}^{t'+1}(M_v(e))$$

(ii) ⇒ (iii) Taking union over all vertices, we get:

$$\forall_{[t..t')\subseteq[T..T+\Delta t)} \{\{\mathcal{C}_t^{t'}(e)\}\}_{e\in\vec{E}} = \bigcup_{v\in V} \{\{\mathcal{C}_t^{t'}(e)\}\}_{e\in\text{in}(v)} = \bigcup_{v\in V} \{\{\mathcal{C}_{t+1}^{t'+1}(e)\}\}_{e\in\text{out}(v)} = \{\{\mathcal{C}_{t+1}^{t'+1}(e)\}\}_{e\in\vec{E}}$$

from which the claim follows.

(iii) ⇒ (iv) Claim follows in a straightforward fashion from the definition of the Φ_i.

(iv) ⇒ (v) Assume that, to the contrary, there exists v' and $e_1, e_2 \in \text{in}(v')$ such that

$$\mathcal{C}_t^{t'}(e_1) - \mathcal{C}_t^{t'}(e_2) \geq 2$$

Let $i = t' - t$ and let us define:

$$c(e_1) = \mathcal{C}_t^{t'}(e_1) - 1$$
$$c(e_2) = \mathcal{C}_t^{t'}(e_2) + 1$$
$$c(e) = \mathcal{C}_t^{t'}(e) \text{ for any other } e$$

We observe, that:

$$\forall_v \sum_{e\in \text{in}(v)} \mathcal{C}_t^{t'}(e) = \sum_{e\in \text{in}(v)} c(e)$$

Thus by the same reasoning as in proof of Lemma 3.7

$$\Phi_i(\mathcal{L}_t) = \sum_v \sum_{e\in \text{out}(v)} \mathcal{C}_t^{t'}(e) > \sum_v \sum_{e\in \text{out}(v)} c(e)^2 = \sum_v \sum_{e\in \text{in}(v)} c(e)^2 \geq \Phi_i(\mathcal{L}_{t+1}),$$

which contradicts the assumption.

(v) $\Rightarrow$ (i) Let us fix v and denote $\text{in}(v) = \{e_1, e_2, \ldots, e_d\}$, $\text{out}(v) = \{f_1, f_2, \ldots, f_d\}$. Under the assumption, we deduce the existence of permutation $(\pi_1, \ldots, \pi_d)$ such that:

$$\forall_{0\leq i\leq \Delta t} \mathcal{C}_T^{T+i}(e_{(\pi_1)}) \geq \mathcal{C}_T^{T+i}(e_{(\pi_2)}) \geq \ldots \geq \mathcal{C}_T^{T+i}(e_{(\pi_d)})$$

because otherwise, we would get that there exists $e_1, e_2, i_1 < i_2 \leq \Delta t$, such that:

$$\mathcal{C}_T^{T+i_1}(e_1) < \mathcal{C}_T^{T+i_1}(e_2), \mathcal{C}_T^{T+i_2}(e_1) > \mathcal{C}_T^{T+i_2}(e_2)$$

implying that

$$\mathcal{C}_{T+i_1}^{T+i_2}(e_1) - \mathcal{C}_{T+i_1}^{T+i_2}(e_2) \geq 2$$

a contradiction with our initial assumption.

There exists a permutation $(\sigma_1), (\sigma_2), \ldots, (\sigma_d)$ such that $f_{(\sigma_1)}, f_{(\sigma_2)}, \ldots, f_{(\sigma_d)}$ have the same cyclic ordering as ρ_v, and $f_{(\sigma_1)}$ is the arc pointed to by $pointer_v$ at beginning of timestep $T+1$, along which the first token at time $T+1$ is propagated. From the cyclic nature of the rotor-router mechanism, we obtain

$$\forall_{0\leq i\leq \Delta t} \mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_1)}) \geq \mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_2)}) \geq \ldots \geq \mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_d)})$$

Thus, the sequences $\left(\mathcal{C}_T^{T+i}(e_{(\pi_1)}), \ldots, \mathcal{C}_T^{T+i}(e_{(\pi_d)})\right)$ and $\left(\mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_1)}), \ldots, \mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_d)})\right)$ are identical since they are sorted sequences of integers, with identical sum and discrepancy not greater than 1 (thus being composed of only two different values).

But $\forall_{0\leq i\leq \Delta t} \mathcal{C}_T^{T+i}(e_{(\pi_j)}) = \mathcal{C}_{T+1}^{T+i+1}(f_{(\sigma_j)})$ implies $\mathcal{L}_T(e_{(\pi_j)}) = \mathcal{L}_{T+1}(f_{(\sigma_j)})$, $\mathcal{L}_{T+1}(e_{(\pi_j)}) = \mathcal{L}_{T+2}(f_{(\sigma_j)})$, $\ldots$, $\mathcal{L}_{T+\Delta t-1}(e_{(\pi_j)}) = \mathcal{L}_{T+\Delta t}(f_{(\sigma_j)})$, meaning that if we pair each of $e_{(\pi_1)}, e_{(\pi_2)}, \ldots$ with corresponding $f_{(\sigma_1)}, f_{(\sigma_2)}, \ldots$ we will obtain a subcycle decomposition. □

Definition 3.10. *We say that a state $\mathcal{H}_t$ admits a* ∞-subcycle decomposition *if it admits i-steps subcycle decomposition for arbitrarily large i.*

Now we proceed to obtain a more algorithmic characterization of stable states:

Theorem 3.11. *Following conditions are equivalent:*

(i) $\mathcal{H}_T$ is stable,

(ii) $\mathcal{H}_T$ admits a ∞-subcycle decomposition,

(iii) $\mathcal{H}_T$ admits a $(2m^2)$-subcycle decomposition.

Proof. *(ii)* $\Rightarrow$ *(i)* If all groups traversing edges follow Eulerian cycles as their routes, then we denote $t_r = \text{lcm}(|\vec{E_1}|, |\vec{E_2}|, \cdots, |\vec{E_p}|)$.

Since $\forall_{e\in\vec{E}} \mathcal{L}_T(e) = \mathcal{L}_{T+t_r}(e)$ by assumption, we easily observe that $\forall_{v\in V} \mathcal{L}_t(v) = \mathcal{L}_{T+t_r}(v)$.

To prove that the states are equivalent, we need to show that the rotor-router pointers for each vertex are in the same position at timesteps T and $T+t_r$. This is equivalent to:

$$\forall_{e_1,e_2\in \text{out}(v)} \mathcal{C}_T^{T+t_r}(e_1) = \mathcal{C}_T^{T+t_r}(e_2)$$

However, if we assume otherwise, then there exist two arcs e_1, e_2 sharing starting points, such that: $|\mathcal{C}_T^{T+t_r}(e_1) - \mathcal{C}_T^{T+t_r}(e_2)| \geq 1$. But that would imply that by taking the cyclic sum twice, $|\mathcal{C}_T^{T+2\cdot t_r}(e_1) - \mathcal{C}_T^{T+2\cdot t_r}(e_2)| \geq 2$ which contradicts the rotor-router property.

$(i) \Rightarrow (iii)$ Let us take T' such that $\mathcal{H}_{T'} = \mathcal{H}_T$ and $T' \geq T + 2m^2$. From Lemma 3.7 we deduce:

$$\forall_{i \geq 0} \Phi_i(\mathcal{H}_T) = \Phi_i(\mathcal{H}_{T+1}) = \Phi_i(\mathcal{H}_{T+2}) = \ldots = \Phi_i(\mathcal{H}_{T'})$$

By using equivalence of clauses (iv) and (i) in Lemma 3.9 , we get the $2m^2$-subcycle decomposition for $\mathcal{H}_T$.

$(iii) \Rightarrow (ii)$ Assume that $\mathcal{H}_T$ which has a $(2m^2)$-subcycle decomposition does not admit a ∞-subcycle decomposition. We denote by τ the largest integer such that $\mathcal{H}_T$ admits a τ-subcycle decomposition ($\tau \geq 2m^2$). By the maximality of τ, taking into account the equivalence of clauses (i) and (v) in Lemma 3.9, there exists a vertex v and two incoming arcs $e_1, e_2 \in \text{in}(v)$ such that:

$$\exists_{T' \in [T..T+\tau]} |\mathcal{C}_{T'}^{T+\tau+1}(e_1) - \mathcal{C}_{T'}^{T+\tau+1}(e_2)| \geq 2. \tag{2}$$

Let $\vec{E}_1, \vec{E}_2$ be the Eulerian cycles from the τ-subcycle decomposition (starting at state $\mathcal{H}_T$), such that $e_1 \in \vec{E}_1, e_2 \in \vec{E}_2$.

Let us denote $\ell = lcm(|\vec{E}_1|, |\vec{E}_2|)$. Observe that $\ell \leq m^2$, since we either have $\vec{E}_1 \cap \vec{E}_2 = \emptyset$, and then $|\vec{E}_1| + |\vec{E}_2| \leq 2m$ implies that $\ell \leq |\vec{E}_1| \cdot |\vec{E}_2| \leq m^2$, or $\vec{E}_1 = \vec{E}_2$, and then $\ell = |\vec{E}_1| \leq 2m \leq m^2$.[†]

Let us observe, that since the groups of tokens are traversing the arcs over the Eulerian subcycles, we have, for any $[t..t+\ell) \subseteq [T..T+\tau]$:

$$\mathcal{C}_t^{t+\ell}(e_1) = \frac{\ell}{|\vec{E}_1|} \sum_{e \in \vec{E}_1} \mathcal{L}_T(e) \tag{3}$$

$$\mathcal{C}_t^{t+\ell}(e_2) = \frac{\ell}{|\vec{E}_2|} \sum_{e \in \vec{E}_2} \mathcal{L}_T(e) \tag{4}$$

Knowing that $2l \leq 2m^2$ and so the cycle decomposition condition also holds for time step $T + 2l$, we have

$$|\mathcal{C}_T^{T+2\ell}(e_1) - \mathcal{C}_T^{T+2\ell}(e_2)| \leq 1$$

and moreover, we can write

$$|\mathcal{C}_T^{T+2\ell}(e_1) - \mathcal{C}_T^{T+2\ell}(e_2)| = \left| \frac{2\ell}{|\vec{E}_1|} \cdot \sum_{e \in \vec{E}_1} \mathcal{L}_t(e) - \frac{2\ell}{|\vec{E}_2|} \cdot \sum_{e \in \vec{E}_2} \mathcal{L}_t(e) \right| = 2 \cdot |\mathcal{C}_T^{T+\ell}(e_1) - \mathcal{C}_T^{T+\ell}(e_2)|.$$

It follows that $\mathcal{C}_T^{T+\ell}(e_1) = \mathcal{C}_T^{T+\ell}(e_2)$, and taking into account (3) and (4) we get that for any $[t..t+\ell) \subseteq [T..T+\tau]$:

$$\mathcal{C}_t^{t+\ell}(e_1) = \mathcal{C}_t^{t+\ell}(e_2) \tag{5}$$

Thus, by invoking (5) we get (with T' chosen as in (2)):

$$|\mathcal{C}_{T'}^{T+\tau+1}(e_1) - \mathcal{C}_{T'}^{T+\tau+1}(e_2)| = |\mathcal{C}_{T'}^{T+\tau+1-\ell}(e_1) + \mathcal{C}_{T+\tau+1-\ell}^{T+\tau+1}(e_1) - \mathcal{C}_{T'}^{T+\tau+1-\ell}(e_2) - \mathcal{C}_{T+\tau+1-\ell}^{T+\tau+1}(e_2)| \overset{(5)}{=}$$

$$\overset{(5)}{=} |\mathcal{C}_{T'}^{T+\tau+1-\ell}(e_1) - \mathcal{C}_{T'}^{T+\tau+1-\ell}(e_2)| \leq 1$$

and we see that τ-subcycle decomposition contradicts with (2). □

A direct result of the proof of Theorem 3.11:

Observation 3.12. *For a stable state $\mathcal{H}_T$, any mapping between incoming and outgoing arcs of v $M_v : \text{in}(v) \to \text{out}(v)$ that is valid with respect to $2m^2$-subcycle decomposition, is valid with respect to ∞-subcycle decomposition.*

[†] $2m \leq m^2$ requires $m \geq 2$, however in graphs with $m = 1$ *every* state is stable.

4 Stabilization time

Theorem 4.1. *State $\mathcal{H}_T$ is stable iff*

$$\sum_{i=1}^{3m} \Phi_i(\mathcal{H}_T) = \sum_{i=1}^{3m} \Phi_i(\mathcal{H}_{T+2m^2}).$$

Proof. $\Rightarrow$ (only if) This is a straightforward result of characterizations from Theorem 3.11 and preservation of potentials (Lemma 3.9, (i) $\Rightarrow$ (iv)).

$\Leftarrow$ (if) By Theorem 3.11 it is enough to prove that $\mathcal{H}_T$ admits a $2m^2$-subcycle decomposition. [†] Let us assume otherwise, that $\mathcal{H}_T$ doesn't admit $2m^2$-subcycle decomposition, and let τ be the largest integer such that there $\mathcal{H}_T$ admits τ-subcycle decomposition ($\tau < 2m^2$). Similarly to the proof of Theorem 3.11, there exists a vertex v and two incoming arcs $e_1, e_2 \in \text{in}(v)$, such that:

$$\exists_{t\in[T..T+\tau]} \left|\mathcal{C}_t^{T+\tau+1}(e_1) - \mathcal{C}_t^{T+\tau+1}(e_2)\right| \geq 2 \tag{6}$$

Let us choose local mapping M valid with respect to the τ-subcycle decomposition (starting at state $\mathcal{H}_T$), and let $\vec{E}_1, \vec{E}_2$ be the Eulerian cycles induced by M, such that $e_1 \in \vec{E}_1, e_2 \in \vec{E}_2$. We will look into the structure of sequences

$$\begin{aligned}(a_0, a_1, \ldots) &= (\mathcal{L}_T(e_1), \mathcal{L}_{T+1}(e_1), \ldots)\\ (b_0, b_1, \ldots) &= (\mathcal{L}_T(e_2), \mathcal{L}_{T+1}(e_2), \ldots)\\ (c_0, c_1, \ldots) &= (a_0 - b_0, a_1 - b_1, \ldots)\end{aligned}$$

Since each separate potential is nondecreasing, we deduce that $\forall_{0\leq i\leq 3m}\Phi_i(\mathcal{H}_T) = \Phi_i(\mathcal{H}_{T+1}) = \ldots = \Phi_i(\mathcal{H}_{T+2m^2})$. Thus, all of $\mathcal{H}_T, \mathcal{H}_{T+1}, \ldots, \mathcal{H}_{T+2m^2-3m}$ admit a $3m$-subcycle decomposition. From this we deduce, that:

$$\forall_{[i..j]\subseteq[0..2m^2)} \text{ if } (j - i < 3m) \text{ then } |c_i + c_{i+1} + \ldots + c_j| \leq 1 \tag{7}$$

(in particular, $c_i \in \{-1, 0, 1\}$ for $0 \leq i < 2m^2$).

From the τ-subcycle decomposition we get that

$$\forall_{[i..j]\subseteq[0..\tau)} |c_i + c_{i+1} + \ldots + c_j| \leq 1$$

and rewriting (6):

$$\exists_{\eta\in[0..\tau]} |c_\eta + c_{\eta+1} + \ldots + c_\tau| \geq 2 \tag{8}$$

Before we proceed further with the proof, we need following structural lemma on the words a, b, c:

Lemma 4.2. *Let $(a_i), (b_i)$ and (c_i) be defined as in Theorem 4.1. If $c_i = 0$ holds for $2m$ consecutive values of i from $0, \ldots, \tau$, then it holds for all of them.*

Proof. $(a_i)_{i=0}^{i\leq\tau}$ and $(b_i)_{i=0}^{i\leq\tau}$ (a and b for short) are periodic sequences with short periods (2m at most). Let $p_1 = |\vec{E}_1|, p_2 = |\vec{E}_2|$ be the periods lengths of those sequences. We want to prove that if they are identical on corresponding fragments, they are equal everywhere. We look into two cases:

- If $\vec{E}_1 = \vec{E}_2$, then $|p_1| = |p_2| \leq 2m$. By a and b being equal on fragment of length $2m$, we get $p_1 = p_2$, giving $a = b$.
- If $\vec{E}_1 \neq \vec{E}_2$, then we get $|p_1| + |p_2| \leq 2m$. By application of Fine and Wilf's theorem (if two periodic words with period lengths x and y are equal on fragment of length $x + y - \gcd(x, y)$, they are identical, see [13]), we get that $a = b$.

□

By Lemma 4.2 we get, that in any sequence that satisfies (8) there cannot be subsequence of $2m$ consecutive values of 0. However, from (7), we deduce that any two occurences of 1 (respectively -1) at distance less than $3m$ have to be separated by at least one occurence of -1 (respectively 1). Thus any occurences of 1 and -1 have to happen in an alternating fashion (separated by arbitrary number of zeroes), leading to contradiction with (8). □

[†] Once again we can eliminate case of $m = 1$ and assume that $2m^2 \geq 3m$.

Round-fair processes To proceed further we intend to provide bounds on the values of i-th potential for rotor-router process (given long enough initialization time). To provide such bounds, we need to analyze the behavior of the cumulated rotor-router processes, *i.e.*, for a fixed Δt, how the distribution of tokens $\mathcal{C}_t^{t+\Delta t}$ evolves with time. Thus, in the following, we will use the broader concept of *round fair processes* denoted by $\mathcal{W}$, as mentioned in Section 2.1.

Definition 4.3. *A process of token distribution (denoted by $\mathcal{W}$) is* round fair*, if*

$$\forall_{e\in\mathrm{out}(v)}\mathcal{W}_t(e) \in \left\{ \left\lfloor \frac{\mathcal{W}_t(v)}{\deg(v)} \right\rfloor, \left\lceil \frac{\mathcal{W}_t(v)}{\deg(v)} \right\rceil \right\} \tag{9}$$

and no tokens are left in nodes:

$$\mathcal{W}_t(v) = \sum_{e\in\mathrm{out}(v)} \mathcal{W}_t(e).$$

We observe that a rotor-router is round fair. Also, due to Observations 3.2, 3.3 and 3.4, for any fixed Δt, cumulated rotor-router in the sense of $\mathcal{W}_t = \mathcal{C}_t^{t+\Delta t}$ also is round fair.

Balancing sets Now we introduce the partition of arcs into *balancing sets*. Arcs belonging to the same set will balance the load over time, while no load balancing happens between different sets. We will formalize those notions in the following.

Definition 4.4. *We define the notion of* distance *between arcs*

$$d(e,e') = \begin{cases} 1 & \text{if } e,e' \in \mathrm{out}(v) \text{ for some } v \\ 1 & \text{if } e,e' \in \mathrm{in}(v) \text{ for some } v \\ p+1 & \text{if } e_1,\ldots,e_p \text{ is shortest sequence that } d(e,e_1)=d(e_1,e_2)=\ldots=d(e_p,e')=1 \\ \infty & \text{otherwise} \end{cases}$$

We say two arcs e, e' belong to the same balancing set $\mathcal{P}$ *iff $d(e,e') < \infty$*

Lemma 4.5. *The set of arcs of any bipartite graph is partitioned into exactly two balancing sets, while for every other graph the partition results in one balancing set. Moreover*

$$\forall_{e,e'\in\mathcal{P}}\ d(e,e') \leq 4D+1$$

and there exists such a path containing at most $2D$ pairs sharing ending vertices and at most $2D+1$ sharing starting vertices.

Proof. Observe that two arcs are in the same balancing set iff there exists path of even length (in terms of edges) connecting the respective starting vertices v of arc e and v' of arc e'. Moreover, the length of such path increased by 1 gives an upper bound on $d(e,e')$. Since in a bipartite graph any path connecting vertices from the same side is of even length, this instantly gives us bound $d(e,e') \leq D+1$ for bipartite graphs when e and e' go in the same direction, and $d(e,e') = \infty$ when they go in opposite directions.

To solve the non-bipartite case, let $\vec{C}$ be the shortest simple cycle of odd length in G (it always exists in non-bipartite graphs). It is easy to show that $|\vec{C}| \leq 2D+1$. Let u be the closest vertex on $\vec{C}$ to v, and u' to v'. Notice that u and u' can be connected by paths of arbitrary parity by using one of two parts of $\vec{C}$ as a connection. Thus, there exists path connecting v to v' going through vertices v, u, u', v' (in this order) of even length at most $4D+1$. Thus the length of the path is at most $4D$.

- If the path doesn't contain both e and e', we get desired bound $d(e,e') \leq 4D+1$ with number of pairs sharing starting vertices bounded by $2D+1$ and pairs sharing ending vertices bounded by $2D$.
- If the path contains at least one of e or e', then $d(e,e') \leq 4D$ giving also bounds on number of pairs of both types being at most $2D$.

□

Now we will proceed to analyze the behavior of the potential defined as in Definition 3.5, with respect to round-fair algorithms. Recall:

$$\Phi(\mathcal{W}_t) \stackrel{\text{def}}{=} \sum_{e \in \vec{E}} (\mathcal{W}_t(e))^2$$

We will denote the value of the potential achieved by distribution of tokens that preserves sums of loads over load balancing sets of arcs (ignoring the restriction that loads are integers) by:

$$\mathcal{B}(\mathcal{W}_t) \stackrel{\text{def}}{=} \sum_{\mathcal{P}} |\mathcal{P}| \cdot \left(\mathop{\mathbf{avg}}_{e \in \mathcal{P}} \mathcal{W}_t(e) \right)^2 \tag{10}$$

where the sum is taken over all possible balancing sets $\mathcal{P}$. Observe, that for non-bipartite graphs (10) reduces to

$$\mathcal{B}(\mathcal{W}_t) = \frac{k^2}{2m} = \text{const}$$

while for bipartite graphs (where k_1 and k_2 are number of tokens in both parts of graph)

$$\mathcal{B}(\mathcal{W}_t) = \frac{k_1^2 + k_2^2}{m} = \text{const}$$

Lemma 4.6.

$$\Phi(\mathcal{W}_t) \geq \mathcal{B}(\mathcal{W}_t)$$

Proof. $\Phi(\mathcal{W}_t) \geq \mathcal{B}(\mathcal{W}_t)$ follows from the convexity of quadratic functions. □

Definition 4.7. *We say that a configuration of tokens $\mathcal{W}_t$ has discrepancy over arcs equal to x, if* $\max_{\mathcal{P}} \max_{e,e' \in \mathcal{P}} (\mathcal{W}_t(e) - \mathcal{W}_t(e')) = x$.

Let us also put the following bound on the potential drop with respect to the discrepancy of token numbers over arcs.

Lemma 4.8. *Let at timestep t a distribution of tokens $\mathcal{W}_t$ has discrepancy over arcs equal to x, for $x > 4D + 1$. Then*

$$\Phi(\mathcal{W}_t) - \Phi(\mathcal{W}_{t+1}) \geq \frac{(x - 4D - 1)(x - 1)}{4D}$$

Proof. From Lemma 4.5 we know, that there exists a sequence of arcs $e_0 = e, e_1, e_2, \ldots, e_{h-1}, e_h = e'$ of length at most $4D + 2$, sharing starting or ending vertices in alternating fashion, such that each vertex belongs to at most 2 of those arcs. Thus

$$\sum_j (\mathcal{W}_t(e_j) - \mathcal{W}_t(e_{j+1})) \geq x$$

Moreover, at most $2D$ of pairs e_j,e_{j+1} share ending vertices and at most $2D + 1$ share starting vertices. Since by (9) loads of arcs sharing starting vertex can differ by at most 1, we get that there exist $I \leq 2D$ vertices $v_1, \ldots, v_I$, such that each vertex v_i admits incoming load discrepancy δ_i (two incoming arcs e_{i1} and e_{i2} such that $\mathcal{W}_t(e_{i1}) - \mathcal{W}_t(e_{i2}) \geq \delta_i$), and that

$$\sum_{i \leq I} \delta_i \geq x - 2D - 1$$

Observe, that:

$$\sum_{e \in \text{in}(v_i)} (\mathcal{W}_t(e))^2 = (\mathcal{W}_t(e_{i1}))^2 + (\mathcal{W}_t(e_{i2}))^2 + \sum_{e \in \text{in}(v_i) \setminus \{e_{i1}, e_{i2}\}} (\mathcal{W}_t(e))^2 \geq$$

$$\geq \left(\mathcal{W}_t(e_{i1}) - \left\lfloor \frac{\delta_i}{2} \right\rfloor \right)^2 + \left(\mathcal{W}_t(e_{i2}) + \left\lfloor \frac{\delta_i}{2} \right\rfloor \right)^2 + \sum_{e \in \text{in}(v_i) \setminus \{e_{i1}, e_{i2}\}} (\mathcal{W}_t(e))^2 \geq \sum_{e \in \text{out}(v_i)} (\mathcal{W}_{t+1}(e))^2$$

which gives

$$\sum_{e\in\text{in}(v_i)}(\mathcal{W}_t(e))^2-\sum_{e\in\text{out}(v_i)}(\mathcal{W}_{t+1}(e))^2\geq(\mathcal{W}_t(e_{i1}))^2+(\mathcal{W}_t(e_{i2}))^2-\left(\mathcal{W}_t(e_{i1})-\left\lfloor\frac{\delta_i}{2}\right\rfloor\right)^2-$$

$$-\left(\mathcal{W}_t(e_{i2})+\left\lfloor\frac{\delta_i}{2}\right\rfloor\right)^2=2\left(\mathcal{W}_t(e_{i1})-\mathcal{W}_t(e_{i2})\right)\left\lfloor\frac{\delta_i}{2}\right\rfloor-2\left\lfloor\frac{\delta_i}{2}\right\rfloor^2\geq 2\left\lceil\frac{\delta_i}{2}\right\rceil\left\lfloor\frac{\delta_i}{2}\right\rfloor\geq\frac{\delta_i^2-1}{2}$$

Summing over all vertices, we get

$$\Phi(\mathcal{W}_t)-\Phi(\mathcal{W}_{t+1})\geq\sum_{i\leq I}\frac{\delta_i^2-1}{2}\geq\left(\sum_{i\leq I}\frac{\delta_i^2}{2}\right)-\frac{I}{2}\geq\frac{(\sum_{i\leq I}\delta_i)^2}{2I}-\frac{I}{2}\geq$$

$$\geq\frac{(x-2D-1)^2}{4D}-D=\frac{(x-4D-1)(x-1)}{4D}$$

□

Lemma 4.9. *If $\mathcal{W}_t$ has discrepancy x, then $\Phi(\mathcal{W}_t)\leq\mathcal{B}(\mathcal{W}_0)+\frac{1}{2}mx^2$.*

Proof. Observe, that by maximizing the sum of squares while preserving the sum and the upper bound on discrepancy, we get that:

$$\sum_{e\in\mathcal{P}}\mathcal{W}_t(e)^2\leq\frac{|\mathcal{P}|}{2}\cdot(\underset{e\in\mathcal{P}}{\mathbf{avg}}(\mathcal{W}_t(e))+x/2)^2+\frac{|\mathcal{P}|}{2}\cdot(\underset{e\in\mathcal{P}}{\mathbf{avg}}(\mathcal{W}_t(e))-x/2)^2$$

Summing above over all $\mathcal{P}$,

$$\Phi(\mathcal{W}_t)\leq\mathcal{B}(\mathcal{W}_0)+\sum_{\mathcal{P}}|\mathcal{P}|\left(\frac{x}{2}\right)^2$$

□

Theorem 4.10. *$\mathcal{W}_T$ has discrepancy of arcs at most $10D$ for $T\geq 16mD\ln k$ steps.*

Proof. Observe, that for discrepancies $x\geq 10D$ we have, by Lemma 4.8

$$\Phi(\mathcal{W}_t)-\Phi(\mathcal{W}_{t+1})\geq\frac{(x-4D-1)(x-1)}{4D}\geq\frac{x^2}{16D}$$

However, by Lemma 4.9

$$x^2\geq 2\frac{(\Phi(\mathcal{W}_t)-\mathcal{B}(\mathcal{W}_0))}{m}$$

Thus,

$$\Phi(\mathcal{W}_{t+1})-\mathcal{B}(\mathcal{W}_0)\leq(\Phi(\mathcal{W}_t)-\mathcal{B}(\mathcal{W}_0))\left(1-\frac{1}{8mD}\right)$$

Let us assume, that after $t\geq 16mD\ln k$ steps the discrepancy is larger than $10D$. Since

$$\Phi(\mathcal{W}_0)-\mathcal{B}(\mathcal{W}_0)\leq\Phi(\mathcal{W}_0)\leq k^2$$

we have

$$\Phi(\mathcal{W}_t)-\mathcal{B}(\mathcal{W}_0)\leq k^2\cdot\left(1-\frac{1}{8mD}\right)^{16mD\ln k}\leq k^2(1/e)^{2\ln k}=1$$

which with respect to Lemma 4.9 contradicts our assumption on the discrepancy. □

We are now ready to prove our main result on the time of stabilization of any rotor-router initial state.

Theorem 4.11. *For any initial state $\mathcal{H}_0$, there exists $T=O(m^4D^2+mD\log k)$ such that $\mathcal{H}_T$ is stable $t_s=O(m^4D^2+mD\log k)$.*

Proof. Let $t_0 = \lceil 16mD\ln(3km)\rceil$. We observe that the cumulated (over Δt rounds) rotor-router process is round-fair, with number of tokens equal to $\Delta t \cdot k$. Observe, that for $t \geq t_0$, $\Delta t \leq 3m$, by Theorem 4.10 the token distribution of $\mathcal{C}_t^{t+\Delta t}$ has discrepancy over arcs at most $10D$, thus

$$\mathcal{B}(\mathcal{C}_0^{\Delta t}) \leq \Phi_{\Delta t}(\mathcal{L}_t) = \Phi(\mathcal{C}_t^{t+\Delta t}) \leq \mathcal{B}(\mathcal{C}_0^{\Delta t}) + 50mD^2.$$

Thus,

$$\sum_{i=1}^{3m} \mathcal{B}(\mathcal{C}_0^i) \leq \sum_{i=1}^{3m} \Phi_i(\mathcal{L}_t) \leq 150m^2D^2 + \sum_{i=1}^{3m} \mathcal{B}(\mathcal{C}_0^i). \tag{11}$$

Let $T > 300m^4D^2 + \lceil 16mD\ln(3km)\rceil$. Let us assume, that $\mathcal{H}_T$ is not stable. Thus, for all $t \in [t_0..T]$,

$$\sum_{i=1}^{3m} \Phi_i(\mathcal{L}_t) - \sum_{i=1}^{3m} \Phi_i(\mathcal{L}_{t+2m^2}) \geq 1$$

and in particular

$$\sum_{i=1}^{3m} \Phi_i(\mathcal{L}_{t_0}) - \sum_{i=1}^{3m} \Phi_i(\mathcal{L}_T) \geq \left\lceil \frac{(T-t_0)}{2m^2} \right\rceil > 150m^2D^2$$

which contradicts with (11). □

5 Lower bound on stabilization time

We now provide a lower bound on the stabilization time of parallel rotor-router walks.

Theorem 5.1. *For any $N, M > 0$, $N \leq M \leq N^2$, there exists an initialization of the rotor router system in some graph with $\Theta(N)$ nodes and $\Theta(M)$ edges such that the stabilization time is $\Omega(M^2 \log N)$.*

Proof. We start the proof by exhibiting for all $n > 0$ an initial configuration with $k = 2$ tokens on the n-node path, for which the stabilization time is $\Omega(n^2 \log n)$. This construction then extends in a straightforward manner to denser graphs.

We will number the nodes of the path with consecutive integers $1, 2, \ldots, n$, starting from its left endpoint. Initially, we assume that both of the tokens are located at node $\lceil n/3\rceil$. Initially, the pointers of all nodes in the range $[2, \lceil n/3\rceil]$ are directed towards the left endpoint of the path, and the pointers of all nodes in the range $[\lceil n/3\rceil + 1, n-1]$ are directed towards the right endpoint; the pointers of the degree-one nodes 1 and n are fixed in their unique position. We assign the tokens with unique identifiers, 1 and 2, defined so that the token labeled 1 is always located not further from the left endpoint of the path than token 2. The identity of a token is persistent over time, and the tokens may only swap identifiers when located at the same node of the path or when simultaneously traversing the same edge of the path in opposite directions.

For the considered initial configuration, the movement of the tokens in the first time steps is such that token 1 is propagated towards the left endpoint of the path (reaching it after $\lceil n/3\rceil - 1$ steps), while token 2 is propagated towards the right endpoint (reaching it after $t_0 = n - \lceil n/3\rceil$ steps). After reaching their respective endpoints, the tokens bounce back and begin to move towards the opposite endpoint of the path. All nodes of the path have now been visited by some token. Following the approach introduced in [17], for all subsequent moments of time t, we consider the partition $V = V_1(t) \cup V_2(t)$ of the set of nodes of the path into so called *domains* $V_1(t)$ and $V_2(t)$, where $V_i(t)$ is the set of nodes of the path whose last visit up to time t inclusive was performed by token i, $i = 1, 2$ (ties on the domain boundary are broken in favor of token 1). As observed in [17], each of the domains $V_i(t)$ is a sub-path of the considered path, containing token i which traverses $V_i(t)$ between its two endpoints, enlarging its domain by one node of the neighboring domain each time it reaches the boundary of the two domains. In our case of $k = 2$ tokens, we define the boundary point $b(t)$ so that $V_1(t) = [1, b(t)]$ and $V_2(t) = [b(t) + 1, n]$. Initially, at the time t_0 when token 2 reaches endpoint n for the first time, we have $b(t_0) = \lceil n/3\rceil$. We will consider the interval of time $[t_0, t_1]$, where t_1 is the first moment of time such that $b(t_1) = \lfloor n/2\rfloor - 1$. Throughout the interval $[t_0, t_1]$, we have $|V_2(t)| > |V_1(t)|$. Thus, for any node v such that $v = b(t)$ for some $t \in [t_0, t_1]$, if token 2 visits v at some time when heading towards the right endpoint of the path, node v will be subsequently visited by token 1 before token 2 returns to v.

It follows that the boundary of the domains can never move to the left by more than one step; formally, for any $t, t' \in [t_0, t_1]$, $t < t'$, we have $b(t') \geq b(t) - 1$.

Now, let $\tau(v)$, for a node $v \in [\lceil n/3 \rceil, \lfloor n/2 \rfloor]$, denote the first moment of time $\tau \in [t_0, t_1]$ such that the domain boundary has reached node v, *i.e.*, $b(\tau) \geq v$. We will show a lower bound on the value of $\tau(\lfloor n/2 \rfloor)$. We begin by lower-bounding the value $\Delta\tau_v = \tau(v+3) - \tau(v)$, for $v \in [\lceil n/3 \rceil, \lfloor n/2 \rfloor - 3]$. Clearly, $\tau(v+3) \geq \tau(v)$. During the considered time interval $(\tau(v), \tau(v+3)]$, the current domain boundary must be visited at least 3 times more by token 1 than by token 2. Since for any $t \in (\tau(v), \tau(v+3)]$, we have $b(t) \geq v - 1$, and the distance between node $v-1$ and the left endpoint of the path is $(v-2)$, the total number of visits of token 1 to the boundary during the considered time interval of length $\Delta\tau_v$ is at most $\lceil \Delta\tau_v / (2(v-2)) \rceil$. On the other hand, since we also have $b(t) \leq v + 3$ during the considered time interval, and the distance between node $v+3$ and the right endpoint of the path is $n-v-3$, we have that the domain boundary is visited by token 2 at least $\lfloor \Delta\tau_v / (2(n-v-3)) \rfloor$ times during the considered time interval. It follows that:

$$\left\lceil \frac{\Delta\tau_v}{2(v-2)} \right\rceil - \left\lfloor \frac{\Delta\tau_v}{2(n-v-3)} \right\rfloor \geq 3,$$

which directly implies:

$$\Delta\tau_v \geq \left(\frac{1}{2(v-2)} - \frac{1}{2(n-v-3)} \right)^{-1} > 2\frac{(v-2)(n-v-3)}{n-2v} > 2\frac{(n/3-2)(n/2-3)}{n-2v} = \frac{(n-6)^2}{6} \cdot \frac{1}{n/2-v}.$$

By summing the time increments $\Delta\tau_v$ over the subpath leading from $\lceil n/3 \rceil$ to $\lfloor n/2 \rfloor$, with a "step" of three nodes, we obtain a lower bound on $\tau(\lfloor n/2 \rfloor)$:

$$\tau(\lfloor n/2 \rfloor) \geq \sum_{a=1}^{n/18-1} \Delta\tau_{\lfloor n/2 \rfloor - 3a} \geq \frac{(n-6)^2}{6} \cdot \sum_{a=1}^{n/18-1} \frac{1}{3a+1} > \frac{(n-6)^2}{18} \sum_{a=2}^{n/18} a^{-1} > \frac{(n-6)^2}{18}(\ln n - 5).$$

Thus, $\tau(\lfloor n/2 \rfloor) = \Omega(n^2 \log n)$. It remains to note that $\tau(\lfloor n/2 \rfloor)$ is a lower bound on the stabilization time of the rotor-router. Indeed, for as long as $t < \tau(\lfloor n/2 \rfloor)$, the domain boundary satisfies $b(t) < \lfloor n/2 \rfloor$, and the left endpoint of the path is visited by token 1 every not more than $2(\lfloor n/2 \rfloor - 2) < n - 1$ steps. On the other hand, in the limit cycle, each node is visited by a token precisely twice in total during $2m = 2(n-1)$ steps, a contradiction. Hence, we obtain a lower bound of $\Omega(n^2 \log n)$ on the stabilization time for a configuration of two tokens on the path.

We now extend our construction to obtain graphs with higher edge density admitting a lower bound of $\Omega(m^2 \log n)$ on stabilization time. For given N and M, we build such a graph G starting with an N-node path P_N, with pointers along internal nodes of the path initialized as before. We then attach the path to a pair of identical cliques on $M' = \Theta(M)$ edges by identifying each endpoint of the clique with one vertex of the respective clique. The initialization of the pointers within the clique is such that a token entering the clique from the path performs a Eulerian walk of exactly $2M'$ arcs on the clique, returning to the path after traversing each edge of the clique exactly twice and returning all pointers within the clique to their initial position. (Such an initialization of pointers on the clique always exists and corresponds to the stable state pointer arrangement of the single-token rotor-router, for the token located at the distinguished vertex.) In our considerations, the initial location of the tokens is assumed to be on node $\lceil N/3 \rceil$ of the clique, as before. The structure of domains within the path is defined analogously as before, and we can disregard the actions of tokens 1 and 2 within their respective cliques completely, replacing them for purposes of analysis by a delay of $2M'$ time steps during which the token is frozen at the left or right endpoint of the path, respectively. Adopting the same notation $\tau(v)$, $v \in [\lceil N/3 \rceil, \lfloor N/2 \rfloor]$, as for the case of the path, we obtain analogously to (5):

$$\left\lceil \frac{\Delta\tau_v}{2M' + 2(v-2)} \right\rceil - \left\lfloor \frac{\Delta\tau_v}{2M' + 2(N-v-3)} \right\rfloor \geq 3,$$

and from there (for $N \geq 9$):

$$\Delta\tau_v \geq \frac{M'^2}{2} \frac{1}{N/2 - v}$$

and finally:

$$\tau(\lfloor N/2 \rfloor) \geq \frac{M'^2}{6}(\ln N - 5).$$

Since $\tau(\lfloor N/2 \rfloor)$ is once again a lower bound on the stabilization time, and since our graph has $\Theta(N)$ and $\Theta(M) = \Theta(M')$ edges, we have obtained the sought example of a rotor router configuration with a stabilization time of $\Omega(M^2 \log N)$. □

6 Simulation of rotor-router

In this section, we will answer the question on how to efficiently query for the state of rotor-router state after given number of steps. First we provide a general solution for parallel rotor-router, followed up by a time efficient algorithm for a single token rotor-router.

6.1 Simulation of the parallel rotor-router

Theorem 6.1. *We can preprocess any $\mathcal{H}_0$, in a polynomial time and space (with respect to $n, m, \log k$) so that we can answer queries of* state *$\mathcal{H}_\tau$ in time $\mathcal{O}(n+m)$.*

Theorem 6.2. *We can preprocess any $\mathcal{H}_0$, in a polynomial time and space (with respect to $n, m, \log k$) so that we can answer queries of $\mathcal{C}_0^\tau(e)$ (total number of visits until timestep τ) in time $\mathcal{O}(n+m)$.*

Proof of Theorem 6.1 and Theorem 6.2. Our first step is to find T such that $\mathcal{H}_T$ is stable. By Theorem 4.11 it is enough to take any $T > 300m^4D^2 + \lceil 16mD \ln(3km) \rceil$. We compute and keep states $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_T$, thus answering any queries of $\mathcal{H}_\tau$ with $\tau < T$ in $\mathcal{O}(n+m)$ time. We store preprocessed $\mathcal{C}_0^\tau(e)$ for any $\tau \in [0..T]$.

By Observation 3.12 we can find any valid ∞-subcycle decomposition of $\mathcal{H}_T$ in polynomial time. By properties of subcycle decomposition we can then find the values of $\mathcal{H}_{T+\tau}(e)$ by finding e' being shifted by τ along the cycle e belongs to. In a similar fashion we find $\mathcal{C}_T^{T+\tau}(e)$ for each arc e, giving us $\mathcal{C}_T^{T+\tau}(v)$ for each vertex v, thus we know the new pointer location for v. Each cycle can be preprocessed with prefix sums such that queries of this type can be answered in $\mathcal{O}(1)$ time, thus giving $\mathcal{O}(n+m)$ time for full $\mathcal{H}_{T+\tau}$ query.

We can preprocess each cycle with prefix sums, thus giving us the access to $\mathcal{C}_T^\tau(e)$ for $\tau \in [T..\infty]$. By adding the value of $\mathcal{C}_0^T(e)$ we get desired $\mathcal{C}_0^\tau(e)$. □

6.2 Simulation of the single token rotor-router

We denote the node where the token starts as v_0, the node the token is located *after* i steps as v_i and the arc the token traverses during the i-th step as $e_i = (v_{i-1}, v_i)$

We partition the sequence of moves into *phases*, where the walk of the i-th *phase*, denoted $\mathcal{E}_i$, ends with a move finishing the i-th *full rotation* of starting node (thus $\mathcal{E}_i$ being an Eulerian cycle for a subgraph of $\vec{G}$ containing all outgoing arcs of v_0). Then the walk of the token can be denoted as

$$\mathcal{E}_\infty = \underbrace{e_1, \ldots, e_{a_1}}_{\mathcal{E}_1}, \underbrace{e_{a_1+1}, \ldots, e_{a_2}}_{\mathcal{E}_2}, \underbrace{e_{a_2+1}, \ldots, e_{a_3}}_{\mathcal{E}_3}, \ldots$$

.

We recall a consequence of main result from [23].

Proposition 6.3. *The sequence of arcs in a phase $i-1$ is always a subsequence of sequence from phase i. Moreover, if some arc is not visited during phase i, the inclusion is strict.*

If we denote by the p index of first phase that covers all arcs of graph (thus $\mathcal{E}_p$ being an Eulerian cycle on all arcs), the Proposition 6.3 can be rephrased as $\mathcal{E}_1 \sqsubset \mathcal{E}_2 \sqsubset \ldots \sqsubset \mathcal{E}_p = \mathcal{E}_{p+1} = \mathcal{E}_{p+2} = \ldots$. For any v, let $\mathrm{expl}(v)$ be the smallest i such that v is visited in the i-th phase. Analogously the smallest i such that $e \in \mathcal{E}_i$ is denoted as $\mathrm{expl}(e)$. We also define $\mathrm{expl}(v_0)$ as 0. We define *the exploratory stage* as maximal consecutive subsequences of $\mathcal{E}_i$ containing arcs not being part of $\mathcal{E}_{i-1}$. The result of Yanovski *et al.* [23], summarized in the next proposition, characterizes the structure of *exploratory stages*.

Proposition 6.4. *The graph induced in each* exploratory stage *is Eulerian. In every phase, each arc is traversed at most once, and furthermore in every phase starting with the* $(\mathrm{expl}(v)+1)$*-th one, every arc adjacent to* v *is traversed exactly once. For every two adjacent nodes* v *and* u *we have that* $|\mathrm{expl}(v)-\mathrm{expl}(u)| \leq 1$*, and actually for every node* w *there exists a sequence of adjacent nodes* $w_0, w_1, w_2, \ldots, w_k$*, where* $w_0 = v_0$ *and* $w_k = w$*, such that* $\mathrm{expl}(w_{i+1}) - \mathrm{expl}(w_i) \in \{0, 1\}$.

To be able to efficiently predict the position of the token at arbitrary time T, we need to efficiently discover the full exploration path $\mathcal{E} = \mathcal{E}_1\mathcal{E}_2 \ldots \mathcal{E}_p$. However, $\mathcal{E}$ might be too large to store it directly (it is of size $\mathcal{O}(m \cdot D)$), therefore we have to store its *compressed form*: the sequence of arcs from $\mathcal{E}_p$ (the Eulerian cycle covering G) together with values of $\mathrm{expl}(e)$ for every arc e.

Algorithm 1 Simulation

```
reset rotor-router pointers
expl(e) ← −1 for every edge e
Q ← [v_0]                                                  // queue with one element
i ← 0
while Q ≠ ∅ do
    i ← i + 1
    R ← ∅                                                  // empty queue
    for all v ∈ Q do
        w ← port(v)      // do a rotor-router walk starting from v until it reaches an already explored arc
        while expl[(v, w)] = −1 do
            expl[(v, w)] = i
            advance(v)                                     // advance the pointer of the node v
            append w to R
            v ← w                                          // proceed to the next node
            w ← port(v)
        end while
    end for
    Q ← R
end while
```

Consider Algorithm 1. Its correctness of Algorithm 1 follows from the fact that in consecutive iterations of the main loop, we simulate traversals of *exploratory stages* from corresponding $\mathcal{E}_i \setminus \mathcal{E}_{i-1}$. In addition to finding the expl value for every arc, it results in the rotor-router pointers being in the same state as after the token performing a full $\mathcal{E}$ walk. We can use this property to find the sequence of arcs from $\mathcal{E}_p$ by simply simulating the rotor-router walk starting from v_0 by another $2 \cdot m$ steps.

Proposition 6.5. *After running Algorithm 1, we can ask queries about position of token after* T *steps in time* $\mathcal{O}(m)$.

To speed up the above method, we denote $\mathcal{E}_p = e'_1, e'_2, \ldots, e'_{2m}$, and let $e'_{2m+1} = e'_1$. Then if we look at the sequence of all values $\mathrm{expl}(e'_1), \mathrm{expl}(e'_2), \ldots, \mathrm{expl}(e'_{2m})$, they have the following property.

Proposition 6.6. *For all* $i = 1, \ldots, 2m$ *we have* $|\mathrm{expl}(e'_i) - \mathrm{expl}(e'_{i+1})| \leq 1$.

We want to find the arc traversed by the token in the T-th step. For this we first check if T exceeds $|\mathcal{E}|$, and if so retrieve the $((T - |\mathcal{E}|) \bmod 2m)$-th element of $\mathcal{E}_p$. Otherwise we need to locate the appropriate $\mathcal{E}_i$ and its element. This reduces to finding the predecessor of T in the set $\{0, |\mathcal{E}_1|, |\mathcal{E}_1| + |\mathcal{E}_2|, \ldots\}$ and then retrieving the $(T - \sum_{j<i} |\mathcal{E}_j|)$-th element of $\mathcal{E}_i$. We will show how to perform both operations in $\mathcal{O}(\log \log m)$ time after linear preprocessing.

Lemma 6.7 (y-fast trees [22])**.** *In the Word RAM model of computation, a set* $S \subseteq [1, U]$ *can be stored in* $\mathcal{O}(|S|)$ *words of space so that we can locate the predecessor of any* x *in* S *using* $\mathcal{O}(\log \log U)$ *time.*

Using the above lemma, the predecessor of T in $\{0, |\mathcal{E}_1|, |\mathcal{E}_1| + |\mathcal{E}_2|, \ldots\}$ can be found in $\mathcal{O}(\log \log m)$ time after linear preprocessing, as $U = \sum_i |\mathcal{E}_i| = \mathcal{O}(mn)$. Therefore we focus on preprocessing all $\mathcal{E}_i$ as to allow

retrieving the k-th element of any $\mathcal{E}_i$ in $\mathcal{O}(\log\log m)$ time. This is not completely trivial, as we want to keep the preprocessing space linear in $|\mathcal{E}_p|$, not $\sum_i |\mathcal{E}_i|$.

Lemma 6.8. *All $\mathcal{E}_i$ can be preprocessed in $\mathcal{O}(m)$ space so that we can retrieve the k-th element of any $\mathcal{E}_i$ in $\mathcal{O}(\log\log m)$ time.*

Proof. We decompose every $\mathcal{E}_i$ into contiguous parts. Every part is a maximal fragment consisting of arcs which are adjacent in the final $\mathcal{E}_p$. In other word, every arc such that its predecessor on $\mathcal{E}_p$ is different than on $\mathcal{E}_i$ begins a new part. Because of Proposition 6.6, the number of such arcs on $\mathcal{E}_i$ is bounded by $|\mathcal{E}_{i+1}|-|\mathcal{E}_i|$. This is because if we have two arcs e, e' which are neighbours on $\mathcal{E}_i$ but not $\mathcal{E}_p$, then there must be at least one arc between them on $\mathcal{E}_{i+1}$. We create a predecessor structure (implemented using Lemma 6.7) storing the positions on $\mathcal{E}_i$ of all such arcs starting a new fragment. Additionally, the structure keeps for every such arc a pointer to the place where the fragment it begins occurs on $\mathcal{E}_p$. Then to retrieve the k-th element of $\mathcal{E}_i$, we use the predecessor structure to find the part the answer belongs to. Then we return the corresponding element of $\mathcal{E}_p$, which takes $\mathcal{O}(1)$ time if we store $\mathcal{E}_p$ in an array. Hence the total query time is $\mathcal{O}(\log\log m)$. To bound the total space, observe that it is equal to $\mathcal{O}(|\mathcal{E}_p| + \sum_i |\mathcal{E}_{i+1}| - |\mathcal{E}_i|) = \mathcal{O}(|\mathcal{E}_p|) = \mathcal{O}(m)$. □

Combining Algorithm 1 and Lemma 6.8, we get following result.

Theorem 6.9. *Graph G can be preprocessed in time $\mathcal{O}(n+m)$ such that we can answer in time $\mathcal{O}(\log\log m)$ to queries of token position after given time T.*

Another natural problem that we would like to solve is how to preprocess $\mathcal{E}$ using small space so that we can efficiently answer queries of the form: *How many times did the token traversed the node v in the first T steps of exploration?*

We will use similar properties as the ones presented above.

Proposition 6.10. *If $vis_i(v)$ is the number of times v is visited during the i-th phase, then $vis_i(v) = 0$ for all $i < \mathrm{expl}(v)$, $vis_i(v) \in (0, \deg(v)]$ for $i = \mathrm{expl}(v)$, and finally $vis_i(v) = \deg(v)$ for all $i > \mathrm{expl}(v)$.*

Theorem 6.11. *Graph G can be preprocessed in time $\mathcal{O}(n+m)$ so that given a node v and a number T we can compute in time $\mathcal{O}(\log\log m)$ how many times node v was visited during the first T steps.*

Proof. We start with performing the preprocessing giving us the compressed representation of $\mathcal{E}$ and expl value of every edge and every node. Then, for every node v we will create a data structure allowing us to efficiently compute how many times v was visited during the first T steps. To do this, for every node v we will keep a triple $(\mathrm{expl}(v), \mathcal{X}_v, \mathcal{Y}_v)$, where $\mathcal{X}_v$ is a predecessor structure with the timestamps of all visits to v that happened in the $\mathcal{E}_{\mathrm{expl}(v)}$-th phase, and $\mathcal{Y}_v$ is a predecessor structure with the timestamps of all visits to v in the $\mathcal{E}_p$-th phase. To process the query, we first apply Theorem 6.9 to determine the position of the token after T steps. Say that in the T-step the token traverses arc e and is in the I-th exploratory phase. If $I < \mathrm{expl}(v)$, then we return 0. If $I = \mathrm{expl}(v)$, then we compute and return the number of elements of $\mathcal{X}_v$ smaller than T, which can be done with a predecessor search. Finally, if $I > \mathrm{expl}(v)$, we know that the total number of times v was visited in all stages up to the $(I-1)$-th is $|\mathcal{X}_v| + (I - \mathrm{expl}(v) - 1) \cdot \deg(v)$. Hence we only need to count the visits to v in the I-th phase. Because in every further phase the order in which the token visits v and traverses e is the same, this can be done by counting how many times v is visited in the $\mathcal{E}_p$-th phase before traversing the arc e. After storing for every arc the time it is traversed in the $\mathcal{E}_p$-th phase, this can be done with a single predecessor search in $\mathcal{Y}_v$.

The size of predecessor structures kept for each node v is $\mathcal{O}(\deg(v))$ and time necessary to create them is $\mathcal{O}(\deg(v))$, thus giving $\mathcal{O}(n+m)$ total preprocessing time and space. □

References

[1] Y. Afek and E. Gafni. Distributed algorithms for unidirectional networks. *SIAM J. Comput.*, 23(6):1152–1178, 1994.

[2] H. Akbari and P. Berenbrink. Parallel rotor walks on finite graphs and applications in discrete load balancing. In Guy E. Blelloch and Berthold Vöcking, editors, *SPAA*, pages 186–195. ACM, 2013.

[3] N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *SPAA*, pages 119–128, 2008.

[4] E. Bampas, L. Gąsieniec, N. Hanusse, D. Ilcinkas, R. Klasing, and A. Kosowski. Euler tour lock-in problem in the rotor-router model. In *DISC*, volume 5805 of *LNCS*, pages 423–435, 2009.

[5] E. Bampas, L. Gąsieniec, R. Klasing, A. Kosowski, and T. Radzik. Robustness of the rotor-router mechanism. In *OPODIS*, volume 5923 of *LNCS*, pages 345–358, 2009.

[6] P. Berenbrink, R. Klasing, A. Kosowski, F. Mallmann-Trenn, and P. Uznański. Improved analysis of deterministic load-balancing schemes. *CoRR*, abs/1404.4344, 2014.

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

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

[9] D. Dereniowski, A. Kosowski, D. Pająk, and P. Uznański. Bounds on the cover time of parallel rotor walks. In E. W. Mayr and N. Portier, editors, *STACS*, volume 25 of *LIPIcs*, pages 263–275. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

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

[11] K. Efremenko and O. Reingold. How well do random walks parallelize? In *APPROX-RANDOM*, pages 476–489, 2009.

[12] R. Elsässer and T. Sauerwald. Tight bounds for the cover time of multiple random walks. *Theor. Comput. Sci.*, 412(24):2623–2641, 2011.

[13] N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the AMS*, 16:109–114, 1965.

[14] A. S. Fraenkel. Economic traversal of labyrinths. *Mathematics Magazine*, 43:125–130, 1970.

[15] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. In *COCOON*, volume 6196 of *LNCS*, pages 130–139, 2010.

[16] L. Gąsieniec and T. Radzik. Memory efficient anonymous graph exploration. In *WG*, volume 5344 of *LNCS*, pages 14–29, 2008.

[17] R. Klasing, A. Kosowski, D. Pająk, and T. Sauerwald. The multi-agent rotor-router on the ring: a deterministic alternative to parallel random walks. In P. Fatourou and G. Taubenfeld, editors, *PODC*, pages 365–374. ACM, 2013.

[18] A. Kosowski and D. Pająk. Does adding more agents make a difference? a case study of cover time for the rotor-router. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 544–555. Springer, 2014.

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

[20] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *FOCS*, pages 694–703, nov 1998.

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

[22] D. E. Willard. Log-logarithmic worst-case range queries are possible in space $\theta(n)$. *Information Processing Letters*, 17(2):81–84, 1983.

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