Speedup in graph exploration with multiple walkers

Dominik Pająk

Includes results of joint work with:

Andrej Ivašković, Dariusz Dereniowski, Ralf Klasing

Adrian Kosowski, Thomas Sauerwald and Przemysław Uznański

Cambridge 22 September 2017

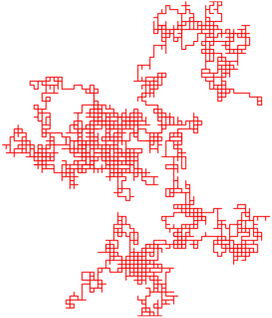
Graph exploration

- A team of **agents** is placed on some subset of nodes of the network.
- The network is an **undirected** graph.
- The agents are propagated along edges of the network following a local set of rules defined for each node.
- The goal of the agents is to visit each node (i.e. to explore the whole network).

Random walk

What is the random walk?

- The agent leaves each node along one of the adjacent links, chosen uniformly at random.
- From the perspective of a node it sends on average the same number of agents in each direction.



Single random walk

Cover time of random walk

Expected time until agent visits all vertices.

Graph class	*Cover time*
Expander, Hypercube, Complete	$\Theta(n \log n)$
2-dim. torus	$\Theta(n \log^2 n)$
Cycle	$\Theta(n^2)$
Lollipop Graph	$\Theta(n^3)$
Any graph	$O(n^3)$, $\Omega(n \log n)$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$
- $H(0, 0) = 2n$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$
- $H(0, 0) = 2n$

We want to compute $H(0, n-1)$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$
- $H(0, 0) = 2n$

We want to compute $H(0, n-1)$

- $H(k-1, k)$ is one less than expected return time of a random walk on a path with $k+1$ vertices starting at the last node,
- $H(k-1, k) = 2k - 1$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$
- $H(0, 0) = 2n$

We want to compute $H(0, n-1)$

- $H(k-1, k)$ is one less than expected return time of a random walk on a path with $k+1$ vertices starting at the last node,
- $H(k-1, k) = 2k - 1$
- $H(i, k) = H(i, k-1) + 2k - 1 = k^2 - i^2$

Single random walk

Consider path on n vertices.

Hitting time

- $H(v, w)$ – expected time to reach v from w

Return time

- $H(v, v) = \frac{1}{\pi_v} = \frac{2m}{d(v)}$
- $H(0, 0) = 2n$

We want to compute $H(0, n-1)$

- $H(k-1, k)$ is one less than expected return time of a random walk on a path with $k+1$ vertices starting at the last node,
- $H(k-1, k) = 2k - 1$
- $H(i, k) = H(i, k-1) + 2k - 1 = k^2 - i^2$
- $H(0, n-1) = (n-1)^2$

Multiple random walks (k = number of agents)

Cover time of multiple random walks

Expected time until every node is visited by some agent.

Speedup

Ratio between the cover time for single walk and for multiple walks.

Graph class	*Speedup*
Expander, Hypercube, Complete, Random	k
Cycle	$\log k$
d-dim. torus ($d > 2$)	$k (k < n^{1-2/d})$

Table: Results from [Elsässer, Sauerwald, 2011] and [Alon, Avin, Koucky, Kozma, Lotker, Tuttle, 2008]

Multiple random walks (k = number of agents)

Cover time of multiple random walks

Expected time until every node is visited by some agent.

Speedup

Ratio between the cover time for single walk and for multiple walks.

Graph class	*Speedup*
Expander, Hypercube, Complete, Random	k
Cycle	$\log k$
d-dim. torus ($d > 2$)	$k (k < n^{1-2/d})$

Table: Results from [Elsässer, Sauerwald, 2011] and [Alon, Avin, Koucky, Kozma, Lotker, Tuttle, 2008]

Conjecture [Alon, Avin, Koucky, Kozma, Lotker, Tuttle, 2008]

Speedup is $O(k)$ and $\Omega(\log k)$ for any graph.

Synergy?

The Markov chain given by Efremenko and Reingold. The cover time for the single random walk equals $\frac{5}{1-\alpha}$, while the cover time for the two random walks starting from any endpoint is $\frac{2.25}{1-\alpha} + o(1/(1-\alpha))$, as $\alpha \to 1$.

Synergy?

The Markov chain given by Efremenko and Reingold. The cover time for the single random walk equals $\frac{5}{1-\alpha}$, while the cover time for the two random walks starting from any endpoint is $\frac{2.25}{1-\alpha} + o(1/(1-\alpha))$, as $\alpha \to 1$.

The speedup is around 2.2

Questions

1. Can it happen for other graphs?
2. Can it happen for any n or any k?
3. Are loop probabilities necessary for this effect?

Questions

1. Can it happen for other graphs?
2. Can it happen for any n or any k?
3. Are loop probabilities necessary for this effect?

We will try to answer 2 and 3.

Result

Theorem

Consider a path with n vertices, where $n \to \infty$. Then the following results hold regardless of the loop-probability of the random walk:

- For $k = 2$, the speed-up satisfies $S_{cov}^{(k)} > 2$.
- For $k \geq 3$, the speed-up satisfies $S_{cov}^{(k)} < k$.

Idea

Lets add more randomness to the random walks!

Idea

Lets add more randomness to the random walks!

Continuous-Time Multiple Random Walks

Let the waiting time between any two transitions of a random walk be independent and identically distributed exponential random variables.

Idea

Lets add more randomness to the random walks!

Continuous-Time Multiple Random Walks

Let the waiting time between any two transitions of a random walk be independent and identically distributed exponential random variables.

Lemma

Continuous-time multiple random walks with waiting time λ is the same as Poisson process with waiting time $k\lambda$ in which in each step a walk chosen uniformly at random makes a move.

Continuous-Time Multiple Random Walks

Lemma

Continuous-time multiple random walks with waiting time λ is the same as Poisson process with waiting time $k\lambda$ in which in each step a walk chosen uniformly at random makes a move.

Continuous-Time Multiple Random Walks

Lemma

Continuous-time multiple random walks with waiting time λ is the same as Poisson process with waiting time $k\lambda$ in which in each step a walk chosen uniformly at random makes a move.

Proof.

Let $T_1, \ldots, T_k$ represent the first times the corresponding walk makes a transition. The first transition by any walk: $T = \min\{T_1, \ldots, T_k\}$

$$\mathbf{Pr}[T \geqslant x] = \mathbf{Pr}[T_1 \geqslant x \cap \ldots \cap T_k \geqslant x] = \mathbf{Pr}[T_1 \geqslant x]^k = e^{-k\lambda x}$$

□

Continuous-Time Multiple Random Walks

Lemma

Continuous-time multiple random walks with waiting time λ is the same as Poisson process with waiting time $k\lambda$ in which in each step a walk chosen uniformly at random makes a move.

Proof.

Let $T_1, \ldots, T_k$ represent the first times the corresponding walk makes a transition. The first transition by any walk: $T = \min\{T_1, \ldots, T_k\}$

$$\mathbf{Pr}[T \geqslant x] = \mathbf{Pr}[T_1 \geqslant x \cap \ldots \cap T_k \geqslant x] = \mathbf{Pr}[T_1 \geqslant x]^k = e^{-k\lambda x}$$

□

Continuous-time model is easier to analyze because:

- Only one walker moves at a single time.
- Loop probabilities (if are the same at each vertex) is simply scaling of the waiting time.

Relating continuous time to discrete time

$t_{\mathrm{cov}}^{(k)}(\vec{u})$ – expected cover time for discrete-time k walks starting at $\vec{u} = (u_1, \ldots, u_k)$

$\widetilde{t_{\mathrm{cov}}}^{(k)}(\vec{u})$ – the same for continuous time

Lemma

For any graph G and $1 \leq k \leq n$,

$$\widetilde{t_{\mathrm{cov}}}^{(k)}(\vec{u}) = \Theta\left(t_{\mathrm{cov}}^{(k)}(\vec{u})\right).$$

Relating continuous time to discrete time

$t_{\text{cov}}^{(k)}(\vec{u})$ – expected cover time for discrete-time k walks starting at $\vec{u} = (u_1, \ldots, u_k)$

$\widetilde{t_{\text{cov}}}^{(k)}(\vec{u})$ – the same for continuous time

Lemma

For any graph G and $1 \leq k \leq n$,

$$\widetilde{t_{\text{cov}}}^{(k)}(\vec{u}) = \Theta\left(t_{\text{cov}}^{(k)}(\vec{u})\right).$$

Lemma (Elsasser and Sauerwald)

If $n^{\epsilon} \leq k \leq n$ for some arbitrary $\epsilon > 0$. Then

$$\mathbf{Pr}\left[t_{\text{cov}}^{(k)}(\vec{u}) \geq \frac{\epsilon}{8} \cdot \frac{n}{k} \cdot \log n\right] \geq 1 - \exp\left(-n^{\epsilon/8}\right).$$

Relating continuous time to discrete time

Lemma

For any graph G and $1 \le k \le n$,

$$\widetilde{t_{\mathrm{cov}}}^{(k)}(\vec{u}) = \Theta(t_{\mathrm{cov}}^{(k)}(\vec{u})).$$

Proof.

- With very high probability the cover time in the discrete model is at least $\log n$ (by Elsasser and Sauerwald).
- If $t > \log n$ we can use the Chernoff bound and Union bound and show that in the continuous model within t steps all the walks make $\Theta(t)$ steps.

□

Gambler's ruin

- Player 1 starts with n_1 coins.
- Player 2 starts with n_2 coins.

Gambler's ruin

- Player 1 starts with n_1 coins.
- Player 2 starts with n_2 coins.
- In each step one player wins a coin from the other one.

Gambler's ruin

- Player 1 starts with n_1 coins.
- Player 2 starts with n_2 coins.
- In each step one player wins a coin from the other one.
- Assume that the game is fair and each player wins in each step with probability $1/2$.

Gambler's ruin

- Player 1 starts with n_1 coins.
- Player 2 starts with n_2 coins.
- In each step one player wins a coin from the other one.
- Assume that the game is fair and each player wins in each step with probability $1/2$.
- What is the time until some of the players will end up having no coins?

Single walk

Gambler's ruin

If there are n coins in total and player 1 starts with k coins, the (fair) game will take on average $k \cdot (n - k)$ rounds.

Single walk

Gambler's ruin

If there are n coins in total and player 1 starts with k coins, the (fair) game will take on average $k \cdot (n - k)$ rounds.

Time to hit an endpoint

- Time to hit the endpoint by a single walk is exactly the time of the Gambler's ruin game.

Single walk

Gambler's ruin

If there are n coins in total and player 1 starts with k coins, the (fair) game will take on average $k \cdot (n-k)$ rounds.

Time to hit an endpoint

- Time to hit the endpoint by a single walk is exactly the time of the Gambler's ruin game.
- It is maximized if we start in the middle and equals $n^2/4$.
- We already computed $H(0, n-1) = n^2$ hence we get
- $t_{cov}^{(1)}(P_n) = \frac{5}{4}n^2$

Multiple walks

Multidimensional Gambler's ruin

- There are k currencies and n coins in total in each currency.

Multiple walks

Multidimensional Gambler's ruin

- There are k currencies and n coins in total in each currency.
- In each step we choose a currency at random and then choose a winner.

Multiple walks

Multidimensional Gambler's ruin

- There are k currencies and n coins in total in each currency.
- In each step we choose a currency at random and then choose a winner.
- We play until some player runs out of coins in any currency.

Multiple walks

Multidimensional Gambler's ruin

- There are k currencies and n coins in total in each currency.
- In each step we choose a currency at random and then choose a winner.
- We play until some player runs out of coins in any currency.

For 2-dimensional game the toal number of steps is approximately $1.178n^2$ (Kmet, Petkovsek)

Theorem

- $t_{cov}^{(2)}(P_n) < 5/8 \cdot n^2$
- $S_{cov}^{(k)} > 2$

Multiple random walks on a d-dimensional grid

Theorem

$d = 2$ / $k \in$	$t_{\text{cov}}^{(k)}$	Speed-up
$[1, \log^2 n]$	$\Theta\left(\frac{n \log^2 n}{k}\right)$	linear
$[\log^2 n, n]$	$\Theta\left(\frac{n}{\log \frac{k}{\ln^2 n}}\right)$	logarithmic

$d \geq 3$ / $k \in$	$t_{\text{cov}}^{(k)}$	Speed-up
$[1, n^{1-2/d} \log n]$	$\Theta\left(\frac{n \log n}{k}\right)$	linear
$[n^{1-2/d} \log n, n]$	$\Theta\left(n^{2/d} / \log\left(\frac{k}{n^{1-2/d} \log n}\right)\right)$	logarithmic

Lower bound for $d = 2$

We want to use the following lemma:

Lemma (Zuckermann 1992)

- $V' \subseteq V$ *s. t.* $|V'| \geq n^{\delta}, \delta > 0$
- *for* $u \in V'$, *at most* $1/n^{\beta}$ *fraction of the* $v \in V'$ *satisfy* $t_{\mathrm{hit}}(u, v) < t$

$$t_{cov} = \Omega(t \cdot \ln n)$$

Lower bound for $d = 2$

We want to use the following lemma:

Lemma (Zuckermann 1992)

- $V' \subseteq V$ *s. t.* $|V'| \geq n^{\delta}, \delta > 0$
- *for* $u \in V'$, *at most* $1/n^{\beta}$ *fraction of the* $v \in V'$ *satisfy* $t_{\mathrm{hit}}(u, v) < t$

$$t_{cov} = \Omega(t \cdot \ln n)$$

We observe that it works for any Markov Chain (not only normal random walk).

Lower bound for $d = 2$

We want to show:

Theorem

On 2-dimensional torus the cover time for $k \in [1, \log^2 n]$ is $\Omega(n \log^2 n/2)$.

Idea

- Lets add more randomness to the random walks!

Lower bound for $d = 2$

We want to show:

Theorem

On 2-dimensional torus the cover time for $k \in [1, \log^2 n]$ is $\Omega(n \log^2 n/2)$.

Idea

- Lets add more randomness to the random walks!
- Consider a process in which k walks are deployed from the origin for a number of steps from geometric distribution with mean $\delta = n \log^2 n/k$.

Lower bound for $d = 2$

We want to show:

Theorem

On 2-dimensional torus the cover time for $k \in [1, \log^2 n]$ is $\Omega(n \log^2 n/2)$.

Idea

- Lets add more randomness to the random walks!
- Consider a process in which k walks are deployed from the origin for a number of steps from geometric distribution with mean $\delta = n \log^2 n/k$.
- Zuckermann's lemma works for such a process (if we manage to satisfy the assumptions).

Lower bound for $d = 2$

We want to show:

Theorem

On 2-dimensional torus the cover time for $k \in [1, \log^2 n]$ is $\Omega(n \log^2 n/2)$.

Idea

- Lets add more randomness to the random walks!
- Consider a process in which k walks are deployed from the origin for a number of steps from geometric distribution with mean $\delta = n \log^2 n/k$.
- Zuckermann's lemma works for such a process (if we manage to satisfy the assumptions).

Lemma

If k walks with geometric length with parameter λ do not cover the graph then $k/2$ walks of length $\lambda/(10c)$ do not cover the graph with probability at least $c/2$.

Lower bound for $d = 2$

- Take V' – set of all vertices at distance at least $1/3 \cdot \sqrt{n}$ to the origin in both dimensions.
- For any $v \in V'$ we can show that if $w \in V'$ satisfies $dist(v, w) \geq n^{49/100}$ then $H(v, w) = \Omega(n \log n)$

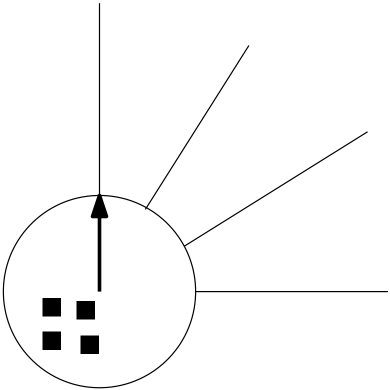
The Rotor-router model

- Each node v has a fixed local port numbering from 1 to $deg(v)$
- The state of each node v is a pointer $p(v) \in \{1, ..., deg(v)\}$.

Rotor-Router Mechanism

For each agent located at node v at the start of time round t:

- The agent is pushed to the neighbor along port $p(v)$
- Pointer $p(v)$ is incremented modulo the degree.



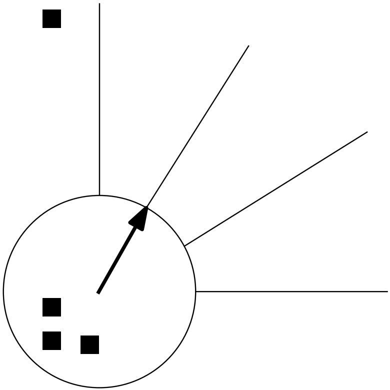
The Rotor-router model

- Each node v has a fixed local port numbering from 1 to $deg(v)$
- The state of each node v is a pointer $p(v) \in \{1, ..., deg(v)\}$.

Rotor-Router Mechanism

For each agent located at node v at the start of time round t:

- The agent is pushed to the neighbor along port $p(v)$
- Pointer $p(v)$ is incremented modulo the degree.



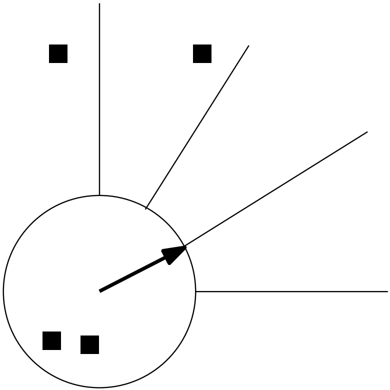
The Rotor-router model

- Each node v has a fixed local port numbering from 1 to $deg(v)$
- The state of each node v is a pointer $p(v) \in \{1, ..., deg(v)\}$.

Rotor-Router Mechanism

For each agent located at node v at the start of time round t:

- The agent is pushed to the neighbor along port $p(v)$
- Pointer $p(v)$ is incremented modulo the degree.



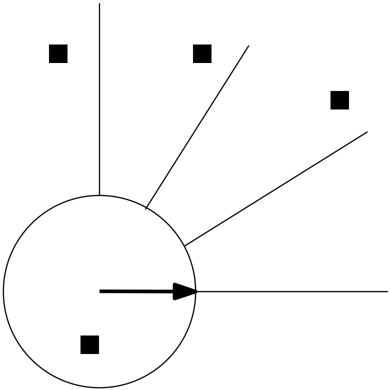
The Rotor-router model

- Each node v has a fixed local port numbering from 1 to $deg(v)$
- The state of each node v is a pointer $p(v) \in \{1, \ldots, deg(v)\}$.

Rotor-Router Mechanism

For each agent located at node v at the start of time round t:

- The agent is pushed to the neighbor along port $p(v)$
- Pointer $p(v)$ is incremented modulo the degree.



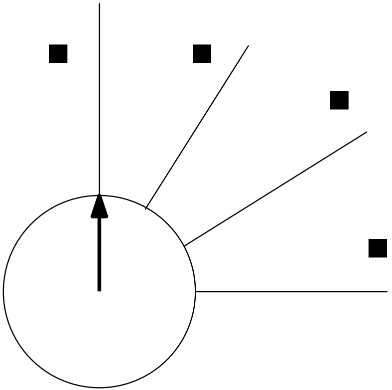
The Rotor-router model

- Each node v has a fixed local port numbering from 1 to $deg(v)$
- The state of each node v is a pointer $p(v) \in \{1, ..., deg(v)\}$.

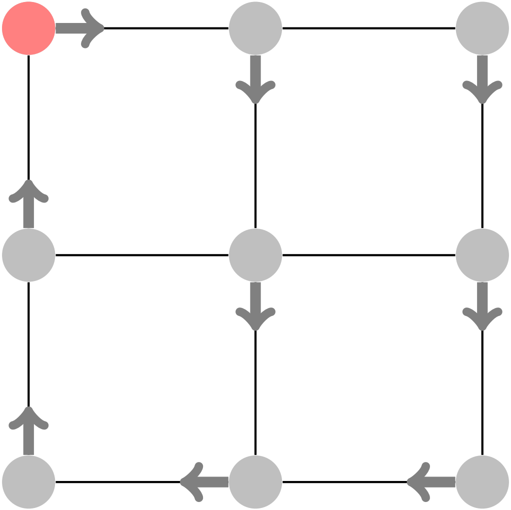
Rotor-Router Mechanism

For each agent located at node v at the start of time round t:

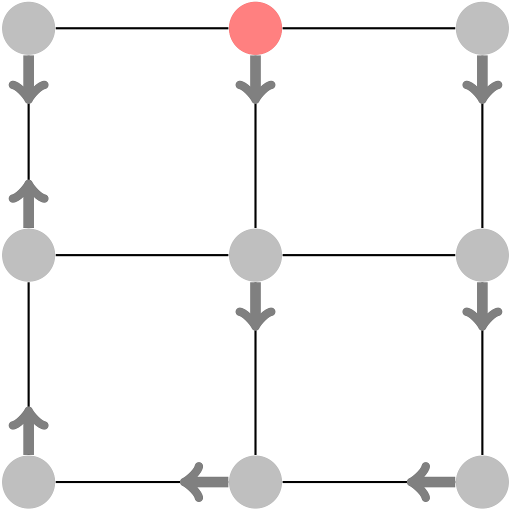
- The agent is pushed to the neighbor along port $p(v)$
- Pointer $p(v)$ is incremented modulo the degree.



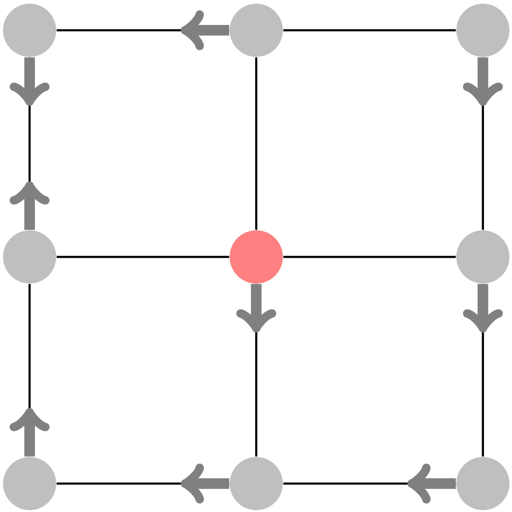
Example (single agent)



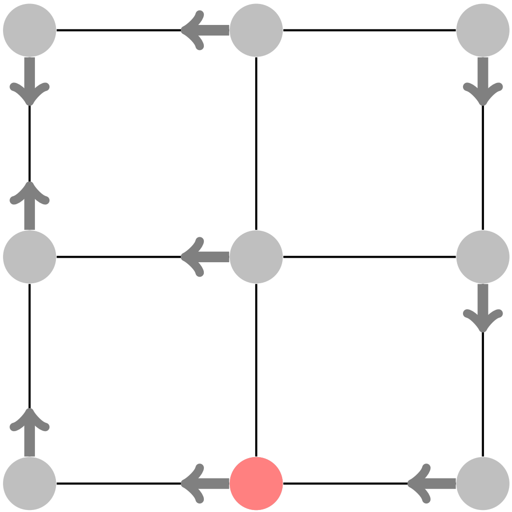
Example (single agent)



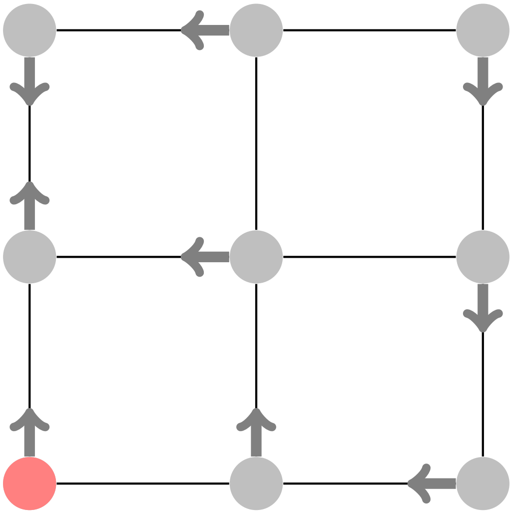
Example (single agent)



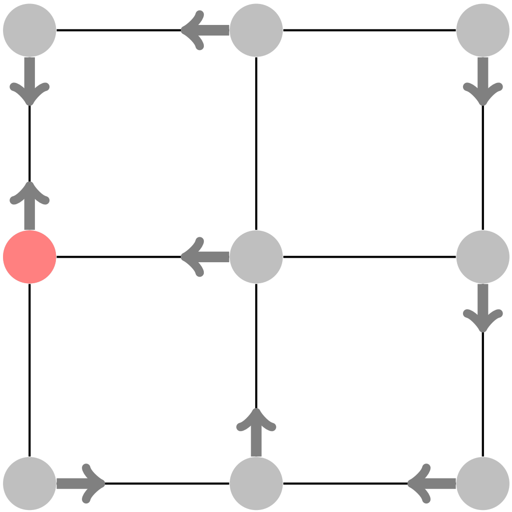
Example (single agent)



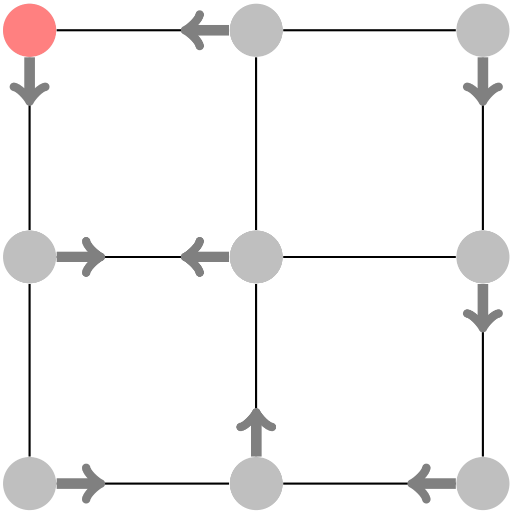
Example (single agent)



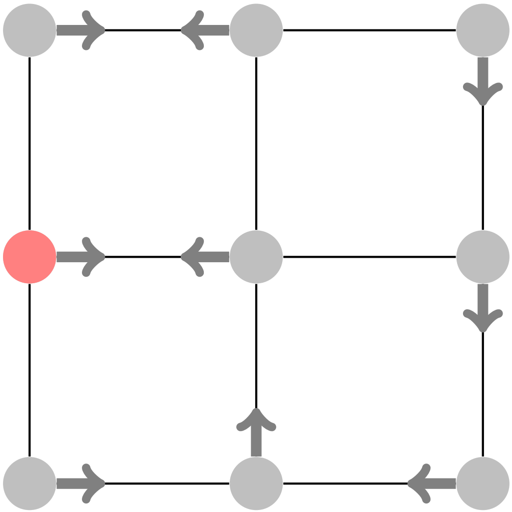
Example (single agent)



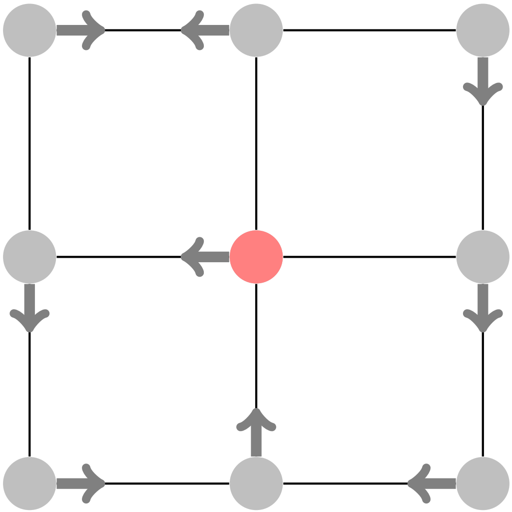
Example (single agent)



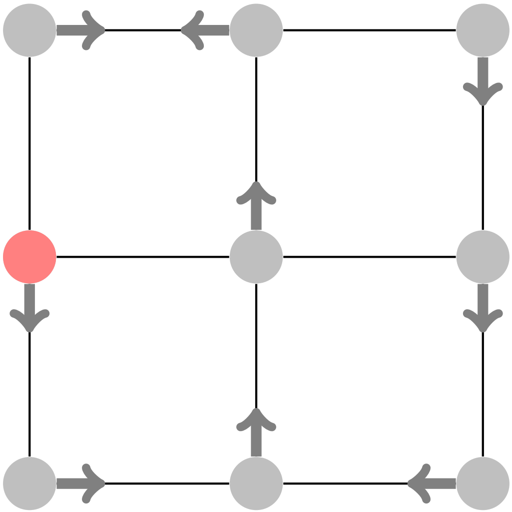
Example (single agent)



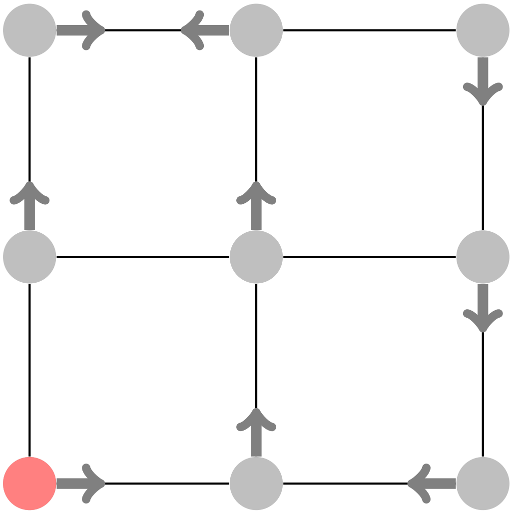
Example (single agent)



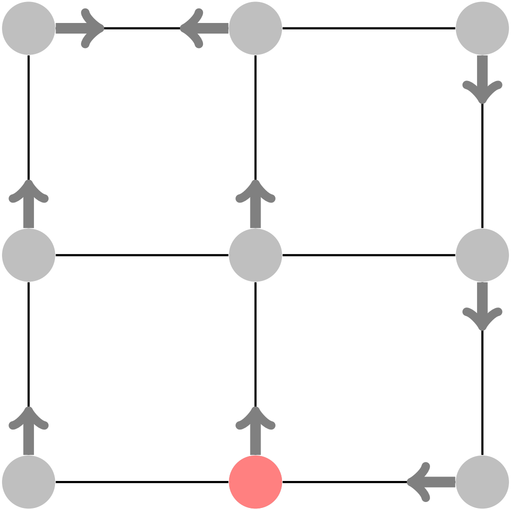
Example (single agent)



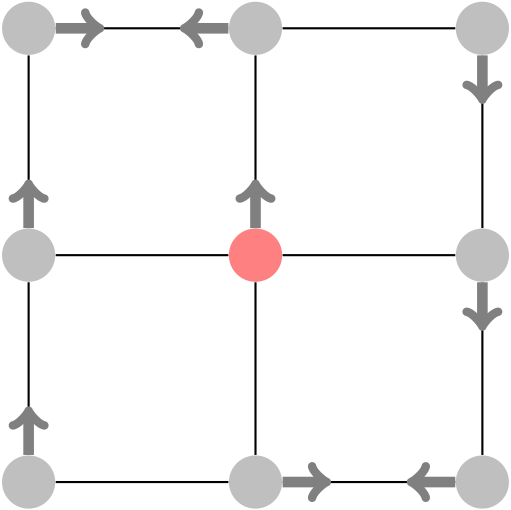
Example (single agent)



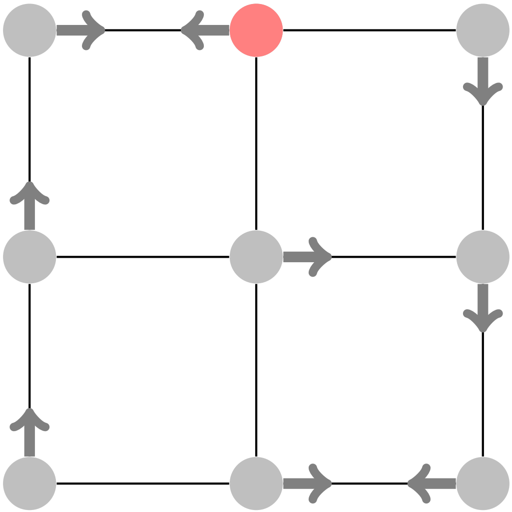
Example (single agent)



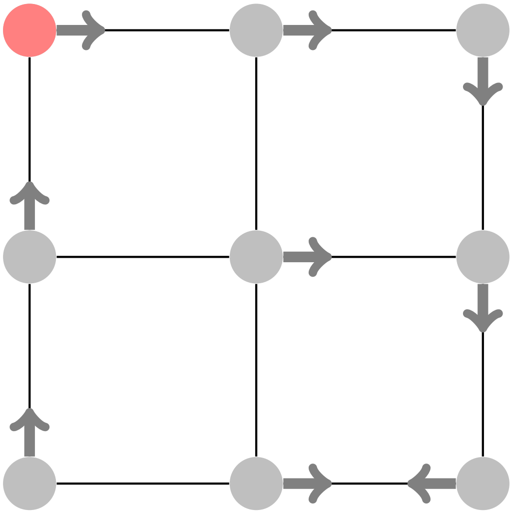
Example (single agent)



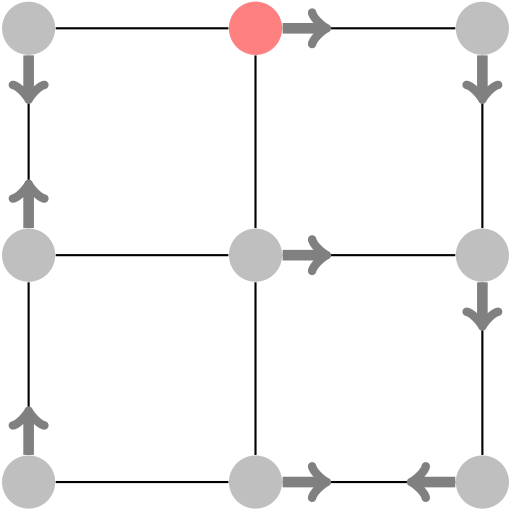
Example (single agent)



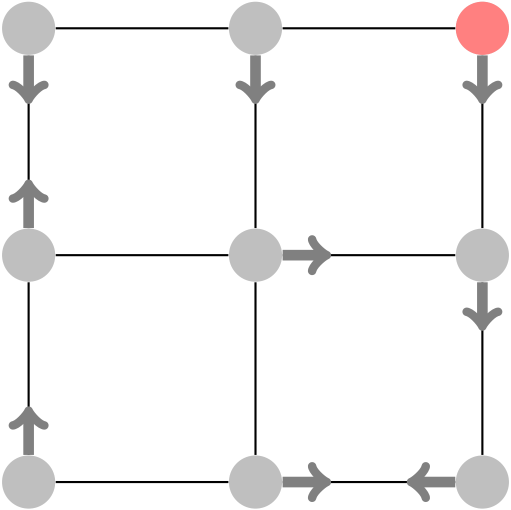
Example (single agent)



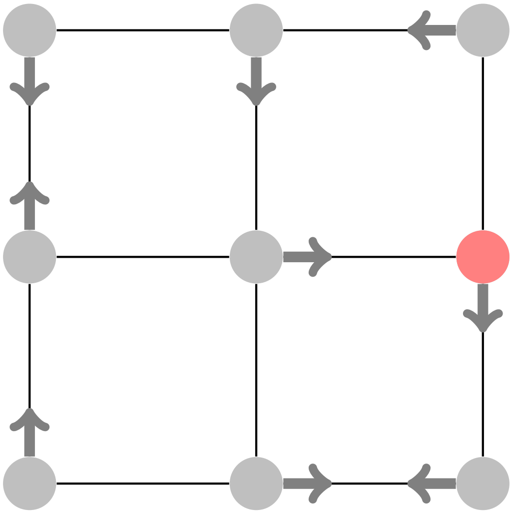
Example (single agent)



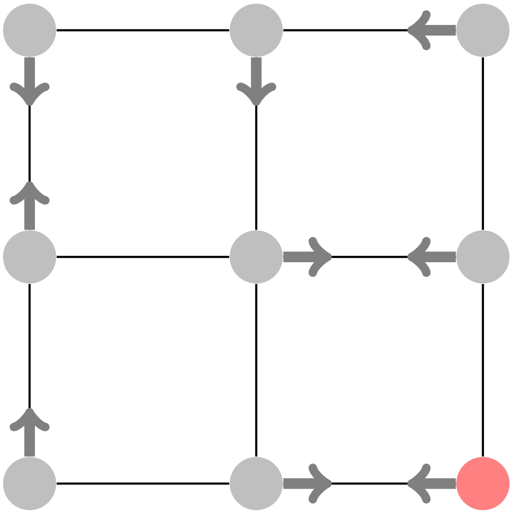
Example (single agent)



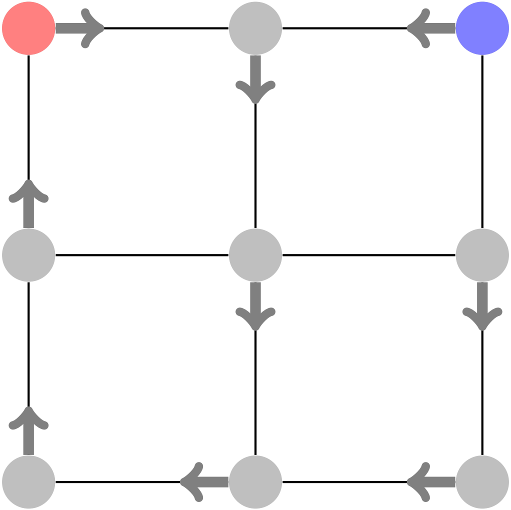
Example (single agent)



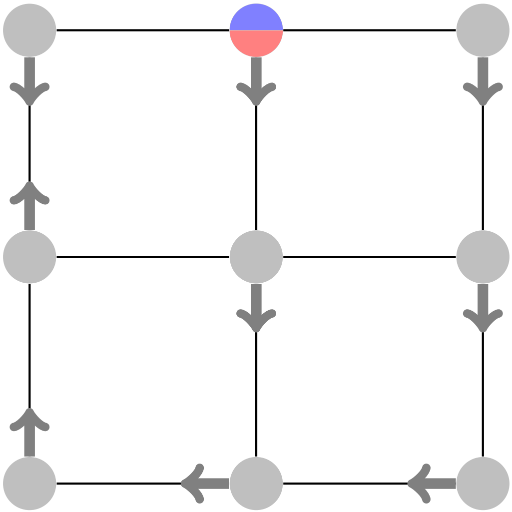
Example (single agent)



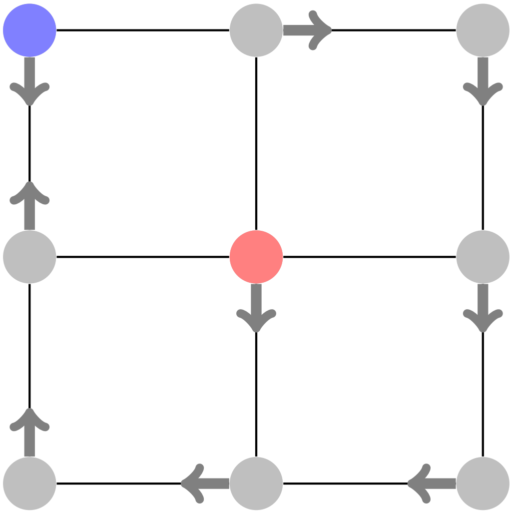
Example (two agents)



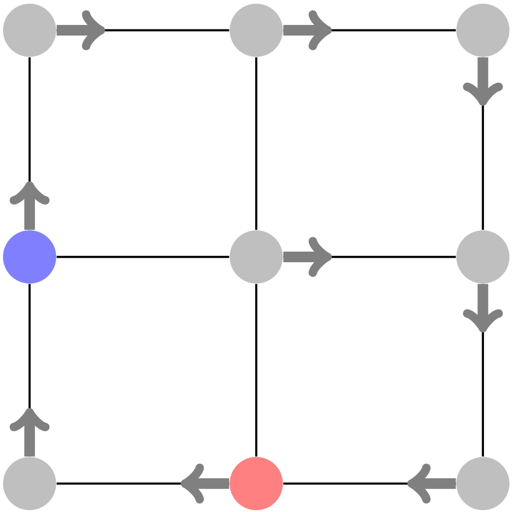
Example (two agents)



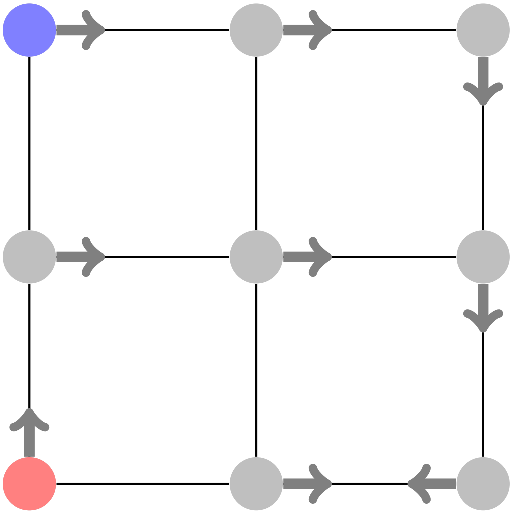
Example (two agents)



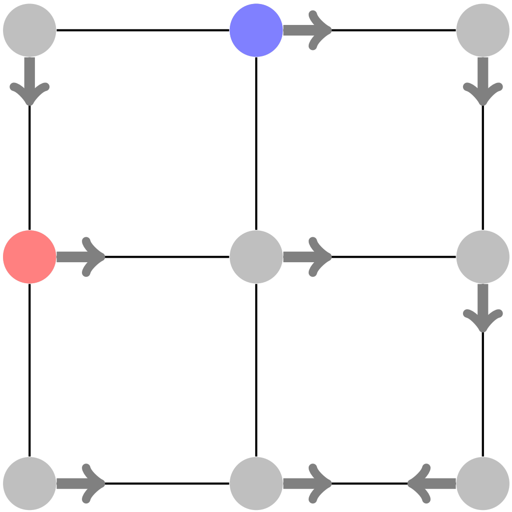
Example (two agents)



Example (two agents)



Example (two agents)



The Rotor-router model

Configuration of the rotor-router

- Initialization of the port numbering
- Initial positions of agents.

When analysing the rotor-router we will always assume the worst possible initial configuration.

Parameters of the rotor-router

Cover time

When will have each node of the graph been reached by some agent, for a worst-case starting configuration?

Lock-in

- The rotor-router is a deterministic process with a finite number of states, hence it must stabilize to a periodic traversal of some cycle in its state space after some initialization phase
- After what time does the rotor-router enter its limit cycle?
- What is the length of the cycle?

Single agent rotor-router

Theorem [Yanovski, Wagner, Bruckstein, 2001]

- For any graph with diameter D and m edges, cover time and lock-in time are bounded by $O(mD)$.
- After this lock-in period, the rotor-router stabilizes to an **Eulerian traversal** of the directed version of the graph (traversing each edge once in each direction).

Theorem [Bampas, Gasieniec, Hanusse, Ilcinkas, Klasing, Kosowski]

- There exists an initial configuration of the rotor-router for which cover time and lock-in time are $\Omega(mD)$.

Single agent rotor-router

Theorem [Yanovski, Wagner, Bruckstein, 2001]

- For any graph with diameter D and m edges, cover time and lock-in time are bounded by $O(mD)$.
- After this lock-in period, the rotor-router stabilizes to an **Eulerian traversal** of the directed version of the graph (traversing each edge once in each direction).

Theorem [Bampas, Gasieniec, Hanusse, Ilcinkas, Klasing, Kosowski]

- There exists an initial configuration of the rotor-router for which cover time and lock-in time are $\Omega(mD)$.

Single agent rotor-router exhibits elegant structural properties.

For any graph, for the worst-case initial configuration

- Cover time is $\Theta(mD)$.
- Lock-in time is $\Theta(mD)$.
- Cycle length is $2m$.

Multi-agent rotor-router

Multiple agents are interacting with the same rotor-router model

- no independence of walks!
- can we have similar results for multi-agent rotor-router as for multiple random walks?

Goal

We want to study the speedup $S(k)$ (a function of k) of the cover time of the multi-agent rotor-router with respect to the single agent.

Multi-agent rotor-router in general graphs

Theorem [Dereniowski, Kosowski, P., Uznanski]

The k-agent rotor-router covers any graph in worst-case time $O(mD/\log k)$ and $\Omega(mD/k)$

- Both of these bounds are achieved for some graph classes.
- The range of speedup for the rotor-router corresponds precisely to the conjectured range of speedup for the random walk.

Outline of argument in proof of $O(mD/\log k)$ cover time.

- Graphs we consider are undirected but it is more convenient to analyse the number of traversals of edges in both directions.

Outline of argument in proof of $O(mD/\log k)$ cover time.

- Graphs we consider are undirected but it is more convenient to analyse the number of traversals of edges in both directions.
- Partition all arcs into possibly empty sets (buckets) E_0, E_1, E_2,..., with an arc e belonging to set E_d at time t if it has been traversed by agents exactly d times up to time t.

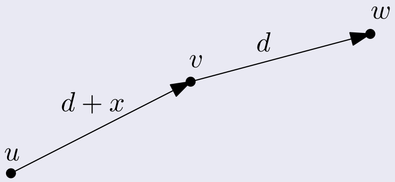
Outline of argument in proof of $O(mD/\log k)$ cover time.

- Graphs we consider are undirected but it is more convenient to analyse the number of traversals of edges in both directions.
- Partition all arcs into possibly empty sets (buckets) E_0, E_1, E_2,..., with an arc e belonging to set E_d at time t if it has been traversed by agents exactly d times up to time t.

Lemma (based on Yanovski 2001)

Suppose that at some moment of time t, there exists a pair of consecutive arcs (u, v) and (v, w), such that

- $(u, v) \in E_{d+x}$,
- $(v, w) \in E_d$.



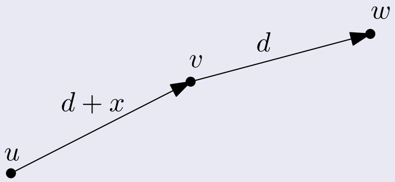
Outline of argument in proof of $O(mD/\log k)$ cover time.

- Graphs we consider are undirected but it is more convenient to analyse the number of traversals of edges in both directions.
- Partition all arcs into possibly empty sets (buckets) E_0, E_1, E_2,..., with an arc e belonging to set E_d at time t if it has been traversed by agents exactly d times up to time t.

Lemma (based on Yanovski 2001)

Suppose that at some moment of time t, there exists a pair of consecutive arcs (u, v) and (v, w), such that

- $(u, v) \in E_{d+x}$,
- $(v, w) \in E_d$.



Then, in step $t + 1$, at least $x - 1$ agents traverse arcs currently belonging to buckets $E_0 \ldots E_d$.

Proof.

- S – the set of nodes which has completed at most d full cycles of if its rotor.

Proof.

- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).

Proof.

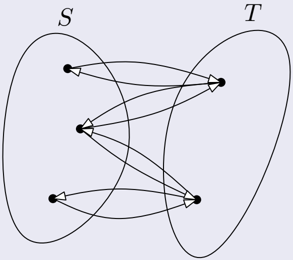
- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.

Proof.

- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.
- Every arc from T to S was traversed at least $d+1$ times.

Proof.

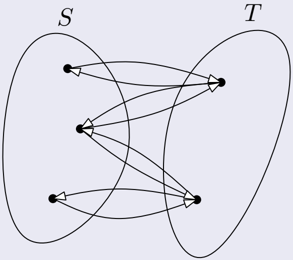
- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.
- Every arc from T to S was traversed at least $d+1$ times.



- There is the same number of edges from S to T as in the opposite direction.

Proof.

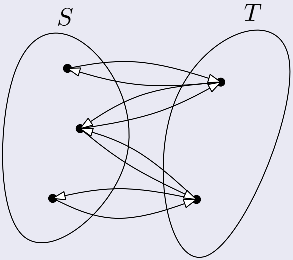
- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.
- Every arc from T to S was traversed at least $d+1$ times.



- There is the same number of edges from S to T as in the opposite direction.
- $u \in T$ and $v \in S$, arc (u, v) from T to S was traversed $d+x$ times.

Proof.

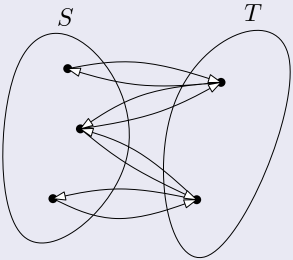
- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.
- Every arc from T to S was traversed at least $d+1$ times.



- There is the same number of edges from S to T as in the opposite direction.
- $u \in T$ and $v \in S$, arc (u, v) from T to S was traversed $d + x$ times.
- "Flow" from T to S is at least $x - 1$.

Proof.

- S – the set of nodes which has completed at most d full cycles of if its rotor.
- T – nodes not belonging to S (completed at least $d+1$ full cycles).
- Every arc from S to T was traversed at most $d+1$ times.
- Every arc from T to S was traversed at least $d+1$ times.



- There is the same number of edges from S to T as in the opposite direction.
- $u \in T$ and $v \in S$, arc (u, v) from T to S was traversed $d+x$ times.
- "Flow" from T to S is at least $x-1$.
- At least $x-1$ agents are in S.

□

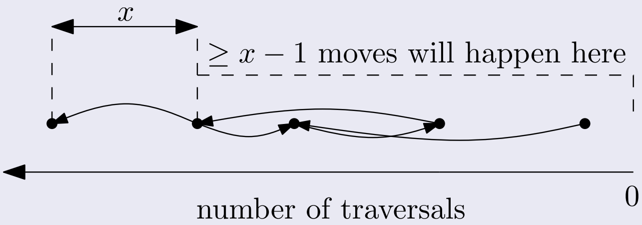
Outline of argument in proof of $O(mD/\log k)$ cover time.

Theorem

The k-agent rotor-router covers any graph in worst-case time $O(mD/\log k)$

1. for $k \le 2^{16D}$,
2. for $k = poly(n)$.

Idea of the argument



Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.
- There exists an arc e with at least $2^b D$ traversals.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.
- There exists an arc e with at least $2^b D$ traversals.
- Take path $\mathcal{P} = \langle e = e_1, e_2, \ldots, e_D \rangle$ of length D starting at e.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.
- There exists an arc e with at least $2^b D$ traversals.
- Take path $\mathcal{P} = \langle e = e_1, e_2, \ldots, e_D \rangle$ of length D starting at e.
- Partition arcs from $\mathcal{P}$ into buckets $I_1, I_2, \ldots I_b$.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.
- There exists an arc e with at least $2^b D$ traversals.
- Take path $\mathcal{P} = \langle e = e_1, e_2, \ldots, e_D \rangle$ of length D starting at e.
- Partition arcs from $\mathcal{P}$ into buckets $I_1, I_2, \ldots I_b$.
- Bucket I_i contains arcs with number of traversals between $2^{i-1}D$ and $2^i D$.

Proof of (1).

- Deploy the system for $2^{b+1}mD/k$ steps, where $b = \log k/2$.
- Total number of visits $= 2^{b+1}mD$.
- There exists an arc e with at least $2^b D$ traversals.
- Take path $\mathcal{P} = \langle e = e_1, e_2, \ldots, e_D \rangle$ of length D starting at e.
- Partition arcs from $\mathcal{P}$ into buckets $I_1, I_2, \ldots I_b$.
- Bucket I_i contains arcs with number of traversals between $2^{i-1}D$ and $2^i D$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Denote by a_i the number of agents currently traversing arcs with number of traversals at most $2^i D$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Denote by a_i the number of agents currently traversing arcs with number of traversals at most $2^i D$.
- "Distance" (difference in the number of traversals) between two consecutive arcs belonging to bucket I_i is at most $a_i + 1$ (by Lemma).

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Denote by a_i the number of agents currently traversing arcs with number of traversals at most $2^i D$.
- "Distance" (difference in the number of traversals) between two consecutive arcs belonging to bucket I_i is at most $a_i + 1$ (by Lemma).
- "Length" of i-th bucket is $2^{i-1} D$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Denote by a_i the number of agents currently traversing arcs with number of traversals at most $2^i D$.
- "Distance" (difference in the number of traversals) between two consecutive arcs belonging to bucket I_i is at most $a_i + 1$ (by Lemma).
- "Length" of i-th bucket is $2^{i-1}D$.
- We get $|I_i| \geq (2^{i-1}D - a_{i-1})/(a_i + 1)$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Denote by a_i the number of agents currently traversing arcs with number of traversals at most $2^i D$.
- "Distance" (difference in the number of traversals) between two consecutive arcs belonging to bucket I_i is at most $a_i + 1$ (by Lemma).
- "Length" of i-th bucket is $2^{i-1} D$.
- We get $|I_i| \geq (2^{i-1} D - a_{i-1})/(a_i + 1)$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b.$
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)
- Right side of this inequality is an arithmetic average.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)
- Right side of this inequality is an arithmetic average.
- For at least half of indices i : $\frac{2^{i-1}}{a_i+1} \leq \frac{18}{b} \Rightarrow a_i \geq \frac{b}{50}2^i$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)
- Right side of this inequality is an arithmetic average.
- For at least half of indices i : $\frac{2^{i-1}}{a_i+1} \leq \frac{18}{b} \Rightarrow a_i \geq \frac{b}{50}2^i$.
- There exists index i^* for which this is true in at least half of all rounds.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)
- Right side of this inequality is an arithmetic average.
- For at least half of indices i : $\frac{2^{i-1}}{a_i+1} \leq \frac{18}{b} \Rightarrow a_i \geq \frac{b}{50}2^i$.
- There exists index i^* for which this is true in at least half of all rounds.
- If we take $\frac{200mD}{b}$ rounds then we accumulate at least $2m2^{i^*}D$ visits in arcs with number of visits at most $2^{i^*}D$.

Outline of argument in proof of $O(mD/\log k)$ cover time.

Proof of (1).

- Buckets are disjoint and contain altogether D elements.
- $D \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i+1} - b$.
- $\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i+1} \leq \frac{1}{b} + \frac{1}{D} \leq \frac{9}{b}$ (Assume: $k \leq 2^{16D} \Rightarrow D \geq \frac{b}{8}$)
- Right side of this inequality is an arithmetic average.
- For at least half of indices $i : \frac{2^{i-1}}{a_i+1} \leq \frac{18}{b} \Rightarrow a_i \geq \frac{b}{50}2^i$.
- There exists index i^* for which this is true in at least half of all rounds.
- If we take $\frac{200mD}{b}$ rounds then we accumulate at least $2m2^{i^*}D$ visits in arcs with number of visits at most $2^{i^*}D$.
- This number of visits is sufficient to "pull" all arcs.

□

Multi-agent rotor-router

Lemma [Yanovski, Wagner, Bruckstein, 2001]

Adding an agent cannot decrease the number of visits at any node at any time. (this implies that $S(k)$ is nondecreasing)

Lemma

Blocking some agents for some time steps cannot increase the number of visits at any node at any time.

Multi-agent rotor-router

Lemma [Yanovski, Wagner, Bruckstein, 2001]

Adding an agent cannot decrease the number of visits at any node at any time. (this implies that $S(k)$ is nondecreasing)

Lemma

Blocking some agents for some time steps cannot increase the number of visits at any node at any time.

Delayed deployments

A process obtained from a rotor-router by defining how many agents to delay at which times and at which nodes.

The slow-down lemma

- $R[k]$ - k-agent rotor router system with an arbitrarily chosen initialization.
- We construct delayed deployment D such that:
  - deployment D explores the graph in at most T steps,
  - in at least τ of these steps all agents were active in D.

Theorem

The cover time $C(R[k])$ of the system can be bounded by:
$\tau \leq C(R[k]) \leq T$.

Applications of the slow-down lemma

The slow-down lemma plays key part in our analysis of the multi agent rotor-router:

- We can analyze $R[k]$ by constructing some easy to analyze, delayed deployment D.
- This allows us to think of the rotor-router as an algorithm, rather than a process which is imposed upon us.
- 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, $C(R[k]) = \Theta(C(D))$.

Outline of argument in proof of $\Omega(mD/k)$ cover time

Theorem

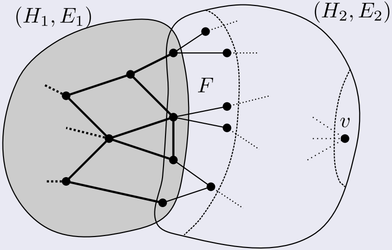
The k-agent rotor-router covers any graph in worst-case time $\Omega(mD/k)$.

Proof (sketch).

- For any graph, we can devise a worst-case initialization of pointers for which there exists a delayed deployment which has some sort of structured behavior,
- using structural lemmas from [Bampas et al. 2009] to decompose the graph into a "heavy" part $H1$ (with many edges) and a "deep" part $H2$ (with large diameter)

Proof (sketch).

- Pointer initialization in $H1$ along an Eulerian circuit in $H1$
- Agents are initially located equidistantly on the circuit.
- Pointer initialization in $H2$ to point towards $H1$



- When any agent leaves "heavy" part and enter "deep" part, we pause all other agents.
- Agent will return to the "heavy" by the same edge it left this part.
- If we contract "heavy" part to one vertex, exploration looks like one-agent exploration of "deep" graph

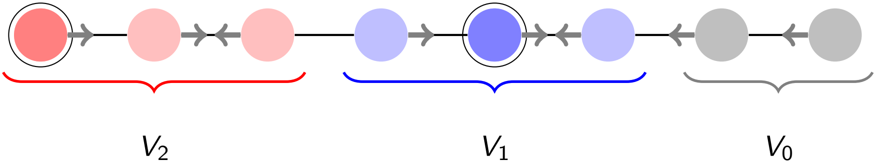
Proof (sketch).

- To explore one level of "deep" part agents need to traverse every edge connecting parts,
- To explore next level agents need to "shift" on the cycle.
- When no agent is in "deep" part then **all** agents are active and walk around the cycle in "heavy" part.
- Total number of steps when all agents are active is $\Theta(mD/k)$.
- We use the slow-down lemma to conclude that undelayed deployment needs time $\Omega(mD/k)$.

□

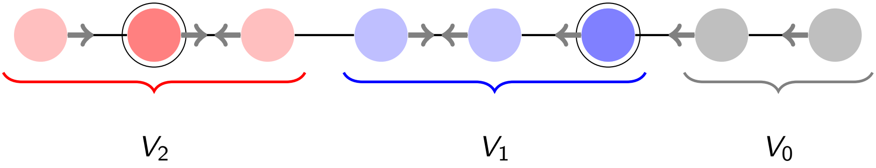
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



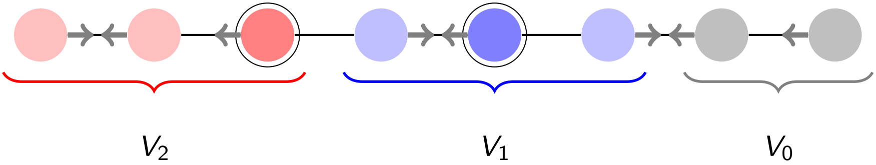
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



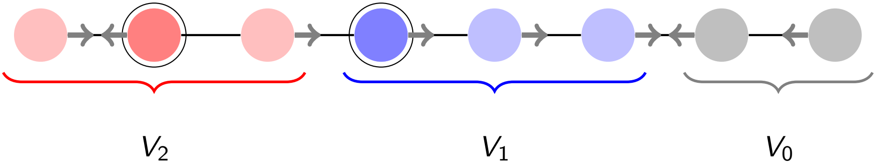
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



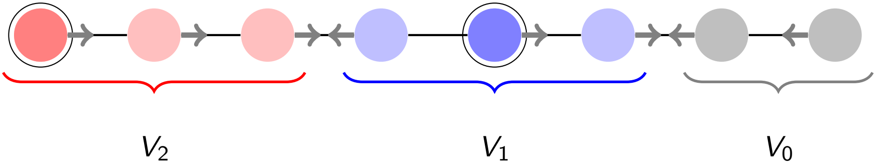
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



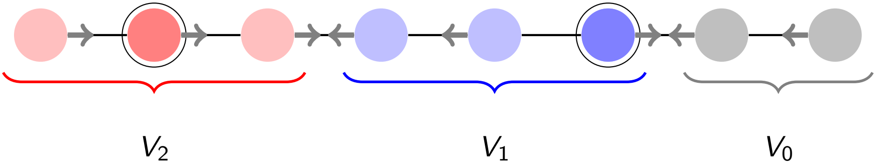
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



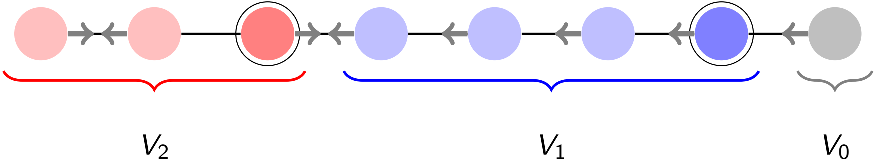
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



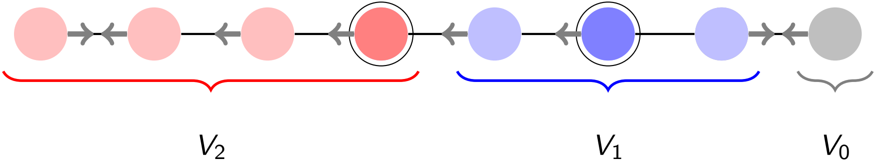
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



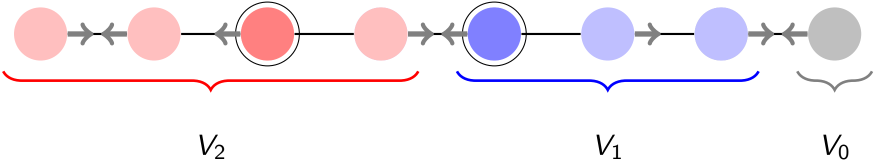
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



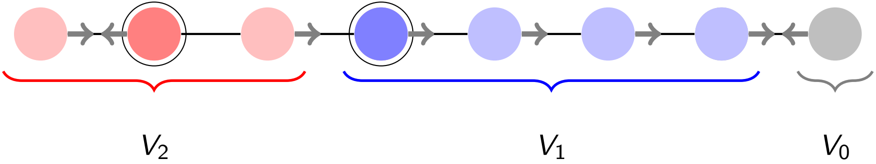
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



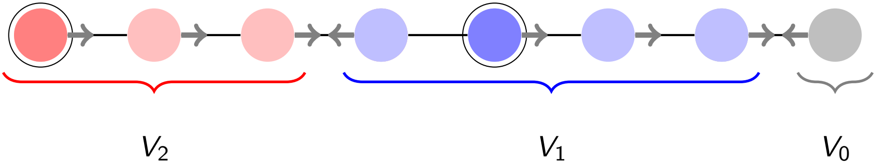
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



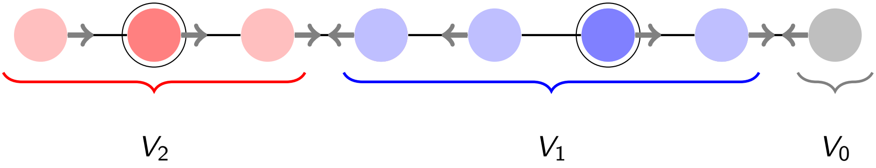
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



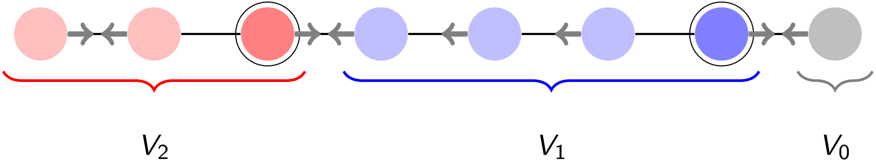
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



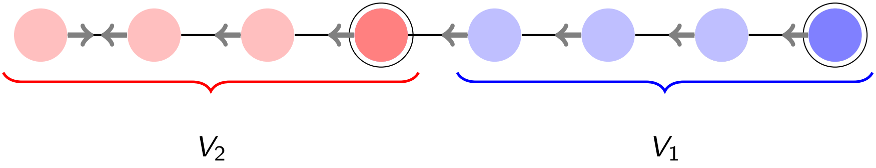
Agent domains

Example on the line, $k = 2$ (starting from some moment...)



Agent domains

Example on the line, $k = 2$ (starting from some moment...)



- Agents are traversing their domains and during each cycle can capture one node from neighboring domain (or at least one node not belonging to any domain).
- Agents with smaller domains will visit borders more frequently thus smaller domains will grow.
- Intuitively the system should converge to domains of equal sizes.

Multi-agent rotor-router on the ring

Theorem

Worst-case cover time for k agent rotor-router on the ring is $\Theta(n^2/\log k)$ when $k < 2^n$.

So the speedup for the ring is $\log k$.

Model	*Cover time* *worst placement*	*best placement*	*Return time*
k-agent rotor-router	$\Theta(n^2/\log k)$	$\Theta(n^2/k^2)$	$\Theta(n/k)$
k random walks (expectations)	$\Theta(n^2/\log k)$ in literature	$\Theta\left(n^2 \Big/ \frac{k^2}{\log^2 k}\right)$	$\Theta(n/k)$ in literature

Discrepancy between the rotor-router and the random walk

To analyse the cover time of the multi agent rotor-router for other graph classes we used a different approach.

Discrepancy in time t

The maximum (taken over all nodes) difference between:

- the **total** number of visits in the k-agent rotor-router,
- the expected **total** number of visits by k random walks,

up to time t.

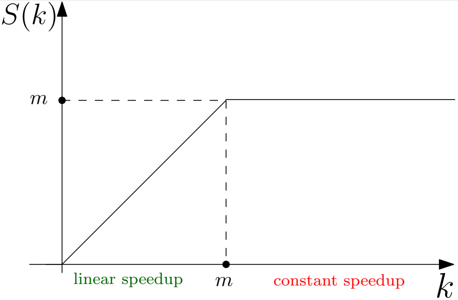
Lemma

- *The discrepancy in time t is bounded by $\Psi_t(G)$*
- $\Psi_t(G) = \max_{v \in V} \sum_{\tau=0}^{t} \sum_{(u_1,u_2) \in \vec{E}} |P_\tau(u_1, v) - P_\tau(u_2, v)|$.
- *$\Psi(G) = \Psi_\infty(G)$ is called **local divergence** and was defined in [Rabani, Sinclair, Wanka 1998].*

Expanders

Using two techniques: delayed deployments and bounded discrepancy we obtained precise asymptotic of the cover time for many graph classes

- Cover time for single agent is $\Theta(mD)$.
- We have linear speedup for k up to m.
- Adding more agents above m gives constant speedup.

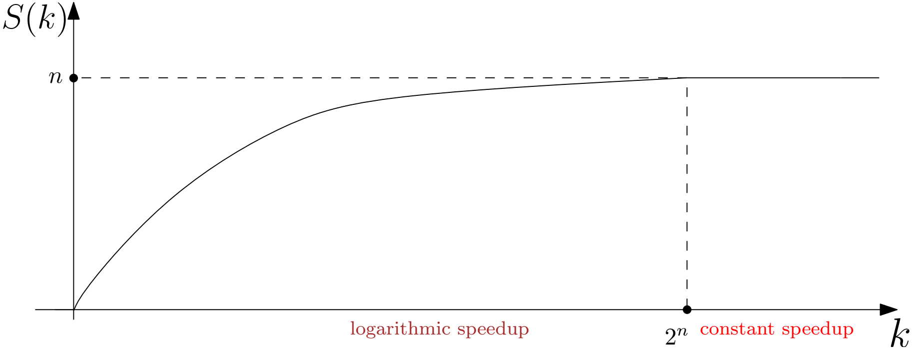


$S(k)$ – speedup (ratio between the cover time for 1 agent and for k agent rotor-router)

- $S(m) = m$ and the cover time for m agents is $\Theta(D)$ (minimum possible).

Cycles

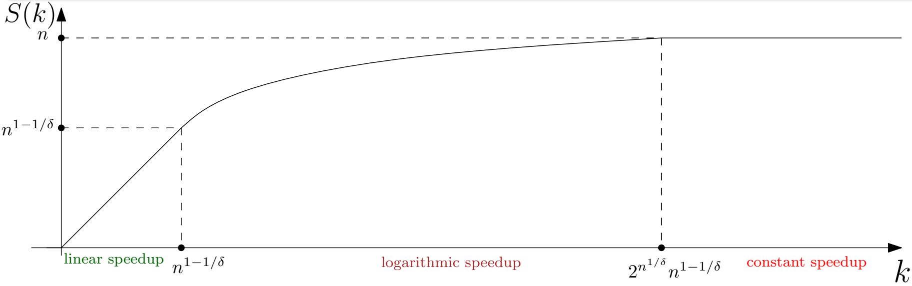
- Cover time for single agent is $\Theta(n^2)$.
- We have logarithmic speedup for k up to 2^n.
- Adding more agents above 2^n gives constant speedup.



- $S(2^n) = n$ and the cover time for 2^n agents is $\Theta(n) = \Theta(D)$ (minimum possible).

δ-dimensional torus

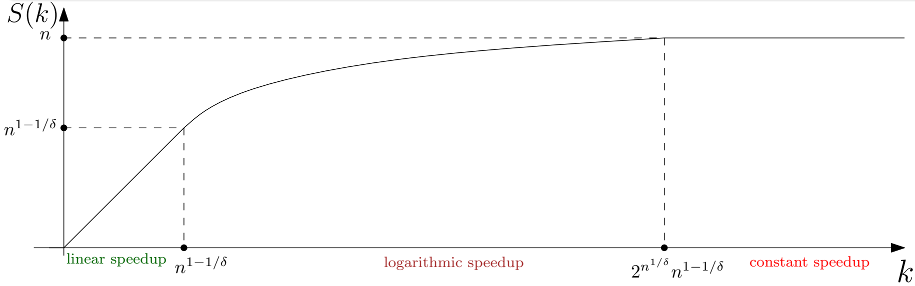
- Cover time for single agent is $\Theta(n^{1+1/\delta})$.
- We have linear speedup for k up to $n^{1-1/\delta}$. (δ-constant)
- Adding more agents above $n^{1-1/\delta}$ gives only logarithmic speedup.



- $S(n^{1-1/\delta}) = n^{1-1/\delta}$ and the cover time is $\Theta(n^{2/\delta})$,
- $S(n^{1-1/\delta} 2^{n^{1/\delta}}) = n$ and the cover time is $\Theta(n^{1/\delta}) = \Theta(D)$.

δ-dimensional torus

- Cover time for single agent is $\Theta(n^{1+1/\delta})$.
- We have linear speedup for k up to $n^{1-1/\delta}$. (δ-constant)
- Adding more agents above $n^{1-1/\delta}$ gives only logarithmic speedup.

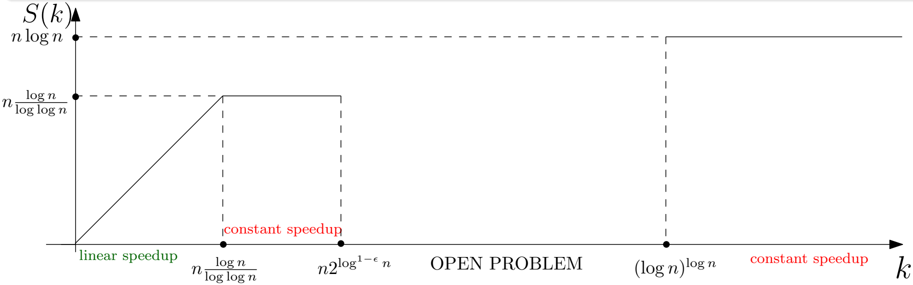


- $S(n^{1-1/\delta}) = n^{1-1/\delta}$ and the cover time is $\Theta(n^{2/\delta})$,
- $S(n^{1-1/\delta} 2^{n^{1/\delta}}) = n$ and the cover time is $\Theta(n^{1/\delta}) = \Theta(D)$.

Team of less than n agents achieves cover time $n^{2/\delta}$ but any team of polynomial size is not sufficient to get cover time $n^{1/\delta}$.

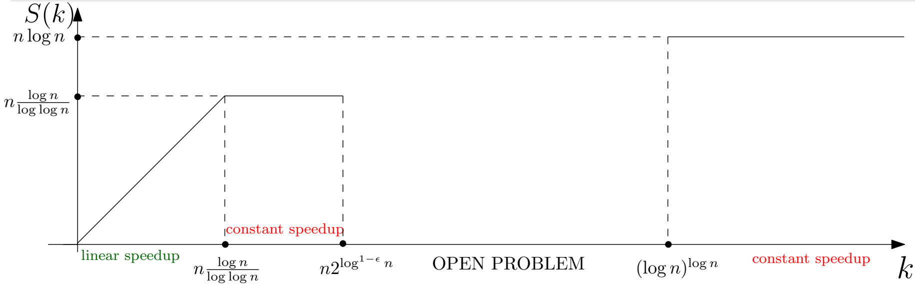
Hypercube

- Cover time for single agent is $\Theta(n \log^2 n)$.
- An interval of linear speedup followed by a period of constant speedup.



- $S(n \frac{\log n}{\log \log n}) = n \frac{\log n}{\log \log n}$, the cover time is $\Theta(\log n \log \log n)$,
- $S((\log n)^{\log n}) = n \log n$, the cover time is $\Theta(\log n) = \Theta(D)$.

- Cover time for single agent is $\Theta(n \log^2 n)$.
- An interval of linear speedup followed by a period of constant speedup.



- $S(n \frac{\log n}{\log \log n}) = n \frac{\log n}{\log \log n}$, the cover time is $\Theta(\log n \log \log n)$,
- $S((\log n)^{\log n}) = n \log n$, the cover time is $\Theta(\log n) = \Theta(D)$.

A similar phenomenon occurs for random walks [Elsässer, Sauerwald, 2011]

There is a period of linear speedup during which the cover time decreases to $\Theta(\log n \log \log n)$ followed by a period of constant speedup.

Multi-agent rotor-router vs. multiple random walks

In terms of the speedup, the multi-agent rotor-router resembles very much multiple random walks.

Graph class	*Speedup (for small k)*	
	Random walk	*Rotor-router*
Cycle	$\log k$	$\log k$
Complete graph	k	k
Star	k	k
Grid $\sqrt{n} \times \sqrt{n}$	k	k
Hypercube	k	k
Binary tree	$\sqrt{k}$	???
General graph	Conjecture: $\Omega(\log k)$	$\Omega(\log k)$
	Conjecture: $O(k)$	$O(k)$

Thank You!