Walking Randomly, Massively, and Efficiently

Jakub Łącki  Slobodan Mitrović  Krzysztof Onak  Piotr Sankowski
Why Random Walks?

• Web ratings [Page, Brin, Motwani, Winograd ‘99] [Berkhin ‘05]  
  [Chierichetti, Haddadan ‘17]

• Graph partitioning [Andersen, Chung, Lang ‘06]

• Random spanning trees [Kelner, Mądry ‘09]

• Laplacian solvers [Andoni, Krauthgamer, Pogrow ‘18]

• Connectivity [Reif ‘85] [Halperin, Zwick ‘94]

• Matching [Goel, Kapralov, Khanna ‘13]

• Property testing [Goldreich, Ron ‘99] [Kaufman, Krivelevich, Ron ‘04]  
  [Czumaj, Sohler ‘10] [Nachmias, Shapira ‘10] [Kale, Seshadhri ‘11]  
  [Czumaj, Peng, Sohler ‘15] [Chiplunkar, Kapralov, Khanna, Mousavifar, Peres ‘18]  
  [Kumar, Seshadhri, Stolman ‘18] [Czumaj, Monemizadeh, Onak, Sohler ‘19]
How to Compute Random Walks?

- Centralized [direct implementation]
- Streaming [Sarma, Gollapudi, Panigrahy ‘11, Jin ‘19]
- Distributed (CONGEST) [Sarma, Nanongkai, Pandurangan, Tetali’13]
- MPC, undirected graphs *(non-independent walks)* [Bahmani, Chakrabarti, Xin ‘11]
How to Compute Random Walks?

- Centralized [direct implementation]
- Streaming [Sarma, Gollapudi, Panigrahy ‘11, Jin ‘19]
- Distributed (CONGEST) [Sarma, Nanongkai, Pandurangan, Tetali’13]
- MPC, undirected graphs (non-independent walks) [Bahmani, Chakrabarti, Xin ’11]

Our result (undirected graphs):
Independent random walks in MPC with sublinear memory per machine.
**Our Results**

*Input:* Undirected graph $G$; length $L$

*Output:* An $L$-length random walk per vertex; walks mutually independent

*Rounds:* $O(\log L)$

*Space per machine:* sublinear in $n$

*Total space:* $O(m L \log n)$. 
Our Results

**Input:** Undirected graph $G$; length $L$

**Output:** An $L$-length random walk per vertex; walks mutually independent

**Rounds:** $O(\log L)$

**Space per machine:** sublinear in $n$

**Total space:** $O(mL \log n)$.

---

**Applications**

- Approximate bipartiteness testing
- Approximate expansion testing
- Approximate connectivity and MST
- PageRank for directed graph
Our Results

**Input:** Undirected graph $G$; length $L$

**Output:** An $L$-length random walk per vertex; walks mutually independent

**Rounds:** $O(\log L)$

**Space per machine:** sublinear in $n$

**Total space:** $O(mL \log n)$.

---

**Applications**

- Approximate bipartiteness testing
- Approximate expansion testing
- Approximate connectivity and MST

**PageRank for directed graph**
Our Results

**Input:** Undirected graph G; length L

**Output:** An $L$-length random walk per vertex; walks mutually independent

**Rounds:** $O(\log L)$

**Space per machine:** sublinear in $n$

**Total space:** $O(mL \log n)$.

**Applications**

- Approximate bipartiteness testing
- Approximate expansion testing
- Approximate connectivity and MST
- PageRank for directed graph

Conditional lower-bound of $\Omega(\log L)$
Random Walks in Undirected Graphs
Random Walks: Doubling by Stitching

**Output**: $\deg(v)$ $L$-length random walk per $v$; walks mutually independent

Track *spare* random walks. Use *spare* to *double* wanted ones.
Random Walks: Doubling by Stitching

Output: $\deg(v)$ L-length random walk per $v$; walks mutually independent

Track spare random walks. Use spare to double wanted ones.
Random Walks: Doubling by Stitching

**Output:** $\deg(v)$ $L$-length random walk per $v$; walks mutually independent

Track *spare* random walks. Use *spare* to double *wanted* ones.
Random Walks: Doubling by Stitching

**Output:** $\deg(v)$ $L$-length random walk per $v$; walks mutually independent.

Track **spare** random walks. Use **spare** to **double** wanted ones.
Random Walks: Doubling by Stitching

**Output:** $\deg(v)$ $L$-length random walk per $v$; walks mutually independent

Track **spare** random walks. Use **spare** to double **wanted** ones.

But how will $w$ know a priori how many walks will pass through it?
Random Walks: Follow Stationary Distribution

But how will we know a priori how many walks will pass through it?
Random Walks: Follow Stationary Distribution

Each vertex $v$ maintains proportionally to $\text{deg}(v)$ random walks.

But how will $w$ know a priori how many walks will pass through it?
Random Walks: Follow Stationary Distribution

Each vertex $v$ maintains proportionally to $\text{deg}(v)$ random walks.

But how will $w$ know a priori how many walks will pass through it?

In expectation, after $t$ steps there are proportionally to $\text{deg}(v)$ walks ending at $v$. 
Random Walks: Takeaway

1. Following stationary distribution allows us to “predict” the future.
Random Walks: Takeaway

1. Following stationary distribution allows us to “predict” the future.

2. The memory requirement is inversely proportional to the min entry of the stationary distribution.

\[ \geq 1/(2m) \]
PageRank for Directed Graphs

**Input:** Directed graph $G^D$

**Output:** $(1+\alpha)$-approximate PageRank;

$\varepsilon$ is the jumping probability

**Rounds:** $\tilde{O}(\varepsilon^{-1} \log \log n)$

**Space per machine:** sublinear in $n$

**Total space:** $\tilde{O}((m + n^{1+o(1)}) \varepsilon^{-4} \alpha^{-2})$. 
Random Walks: Undirected vs Directed

Undirected graphs

VS

Directed graphs
(Prelude) Random Walks: Undirected vs Directed

**Undirected graphs**

- Stationary distribution is easy to compute: $\frac{\text{deg}(v)}{2m}$.

**Directed graphs**

- Stationary distribution of $v$ is "nicely" lower-bounded.
(Prelude) Random Walks: Undirected vs Directed

**Undirected graphs**

- Stationary distribution is easy to compute: $\frac{\text{deg}(v)}{2m}$.
- Stationary distribution of $v$ is “nicely” lower-bounded.

**Directed graphs**

- Stationary distribution can be difficult to compute.

VS
(Prelude) Random Walks: Undirected vs Directed

Undirected graphs

- Stationary distribution is easy to compute: $\deg(v) / (2m)$.
- Stationary distribution of $v$ is “nicely” lower-bounded.

Directed graphs

- Stationary distribution can be difficult to compute.
- Stationary distribution of $v$ can be $O(1/2^n)$.
PageRank: Undirected vs Directed Graphs

**Input:**

\[ P = GD^{-1} \]

\[ T = (1 - \epsilon)P + \frac{\epsilon}{n}11^T \]

**Output:** Stationary distribution of \( T \)
PageRank: Undirected vs Directed Graphs

\[ \text{Input: } P = GD^{-1} \]

\[ T = (1 - \epsilon)P + \frac{\epsilon}{n} \mathbf{11}^T \]

\[ \text{Output: Stationary distribution of } T \]

- Following \( P \) with prob. \( 1 - \epsilon \).
- Jumping to a random vertex.
- Walk matrix of \( G \).
PageRank: Undirected vs Directed Graphs

**Input:** \( P = GD^{-1} \)

\[
T = (1 - \epsilon)P + \frac{\epsilon}{n}11^T
\]

**Output:** Stationary distribution of \( T \)

PageRank can be approximated from random walks of \( T \). [Breyer ‘02]
PageRank: Undirected vs Directed Graphs

**Input:** \( P = GD^{-1} \)

\[ T = (1 - \epsilon)P + \frac{\epsilon}{n}11^T \]

**Output:** Stationary distribution of \( T \)

PageRank can be approximated from random walks of \( T \). [Breyer '02]

Undirected graphs

Directed graphs

\( T \) and \( P \) are "similar".

VS
PageRank: Undirected vs Directed Graphs

**Input:** \( P = GD^{-1} \)

\[
T = (1 - \epsilon)P + \frac{\epsilon}{n} \mathbb{1} \mathbb{1}^T
\]

**Output:** Stationary distribution of \( T \)

PageRank can be approximated from random walks of \( T \). [Breyer ‘02]

**Undirected graphs**

\( T \) and \( P \) are “similar”.

**Directed graphs**

We do not know stationary distribution of \( T \).
PageRank: Undirected vs Directed Graphs

**Input:** $P = GD^{-1}$

$T = (1 - \epsilon)P + \frac{\epsilon}{n} \mathbf{1} \mathbf{1}^T$

**Output:** Stationary distribution of $T$

PageRank can be approximated from random walks of $T$. [Breyer ‘02]

Undirected graphs

$T$ and $P$ are “similar”.

Directed graphs

We do not know stationary distribution of $T$.

Stationary distribution of $v$ w.r.t. to $P$ can be $O(1/2^n)$.

Stationary distribution of $v$ w.r.t. $T$ at least $\epsilon/n$. 

VS
Improvise. Adapt. Overcome
PageRank: **Molding Undirected to Directed**

- PageRank for undirected $G$.
- PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

PageRank for undirected $G$.

PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

Randome walks for $(1-\delta)G + \delta G^D$.

PageRank for undirected $G$.

PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

Random walks for $(1-\delta)G + \delta G^D$.

PageRank can be approximated from random walks of $T$. [Breyer ‘02]

PageRank for undirected $G$.

PageRank for $(1-\delta)G + \delta G^D$.

PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

PageRank can be approximated from random walks of $T$. [Breyer ‘02]

PageRank for undirected $G$.

PageRank for $(1-\delta)G+\delta G^D$.

PageRank for $(1-2\delta)G+2\delta G^D$.

PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

Random walks for $(1-\delta)G+\delta G^D$.

PageRank can be approximated from random walks of $T$. [Breyer ‘02]

PageRank for undirected $G$.
PageRank for $(1-\delta)G+\delta G^D$.
PageRank for $(1-2\delta)G+2\delta G^D$.
... 
PageRank for $\delta G+(1-\delta)G^D$.
PageRank for directed $G^D$. 
PageRank: Molding Undirected to Directed

“Small” changes in $T$ require a “small” increase in the number of spare walks.

Random walks for $(1-\delta)G+\delta G^D$.

PageRank can be approximated from random walks of $T$. [Breyer ‘02]

PageRank for undirected $G$.

PageRank for $(1-\delta)G+\delta G^D$.

PageRank for $(1-2\delta)G+2\delta G^D$.

PageRank for $\delta G+(1-\delta)G^D$.

PageRank for directed $G^D$. 
PageRank: Takeaway

1. The stationary distribution is lower-bounded by $\epsilon/n$. 
PageRank: Takeaway

1. The stationary distribution is lower-bounded by $\varepsilon/n$.

2. “Small” changes in a walk matrix affect the stationary distribution by little.
Massively Parallel Computation (MPC) round

Data: 

$N$ machines:
Massively Parallel Computation (MPC) round

Data:  

\( N \) machines:
Massively Parallel Computation (MPC) round

Data:

\( N \) machines:

process data \textit{locally}
Massively Parallel Computation (MPC) round

Data:

$N$ machines:

Next-round data:
Massively Parallel Computation (MPC) round

Data:

$N$ machines:

Next-round data:

One round
Massively Parallel Computation (MPC) round

Data:

$N$ machines:

Next-round data:

One round
Massively Parallel Computation (MPC) parameters

\[ N = \# \text{ of machines} \]
\[ S = \text{space per machine} \]
For graphs, \[ N \times S = \Theta(\# \text{ of edges}) \]
Massively Parallel Computation (MPC) parameters

\[ N = \# \text{ of machines} \]
\[ S = \text{space per machine} \]
For graphs, \[ N \times S = \Theta(\# \text{ of edges}) \]

Interesting case:
\[ S \text{ much smaller than the input size} \]

Goal:
\[ \text{make the small \# of rounds} \]