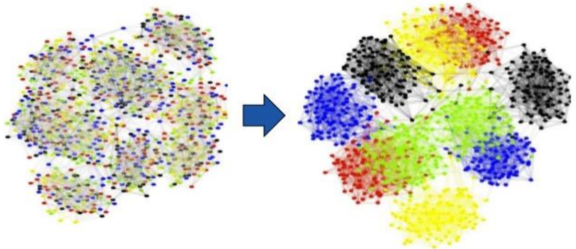
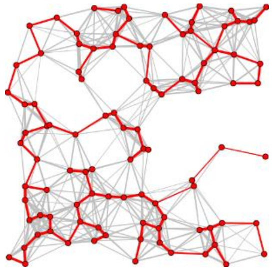
Walking Randomly, Massively, and Efficiently

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(m L 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(m L 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(m L log n).

Conditional lower-bound of Ω(log L)

Applications

Approximate bipartiteness testing

Approximate expansion testing

Approximate connectivity and MST

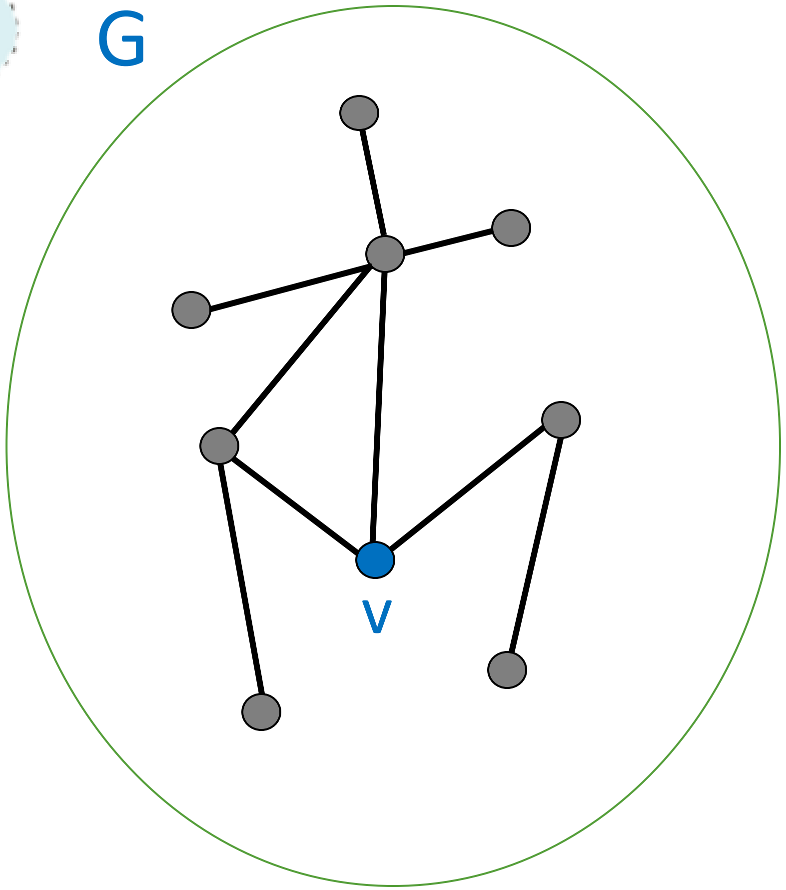
PageRank for **directed** graph

Random Walks in Undirected Graphs

Random Walks: Doubling by Stitching

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

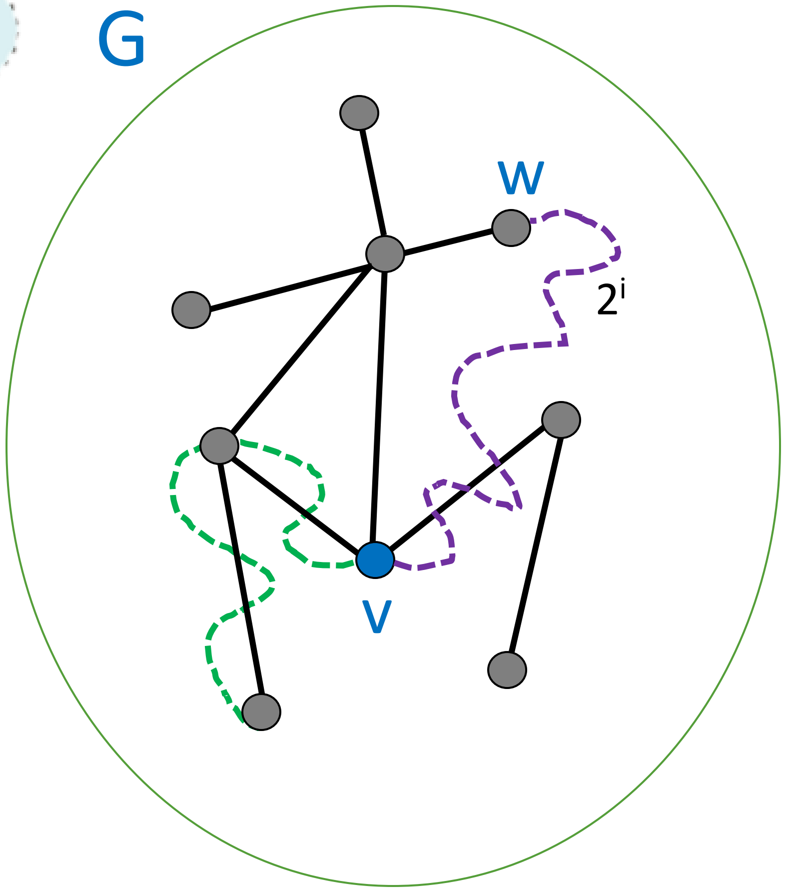
Track spare random walks. Use spare to **double** wanted ones.



Random Walks: Doubling by Stitching

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

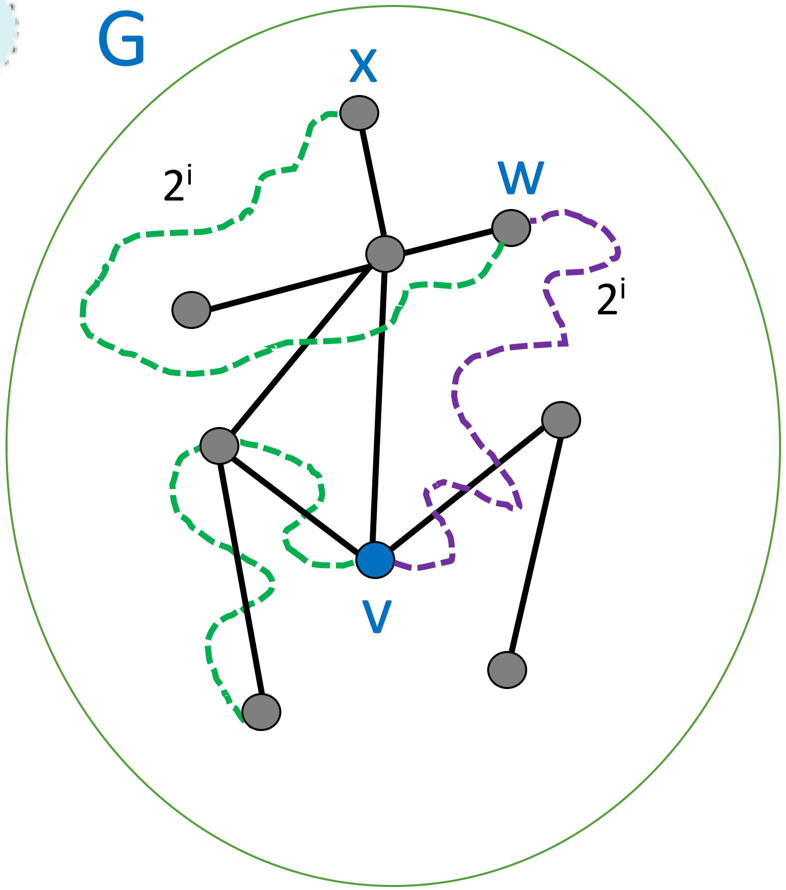
Track spare random walks. Use spare to **double** wanted ones.



Random Walks: Doubling by Stitching

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

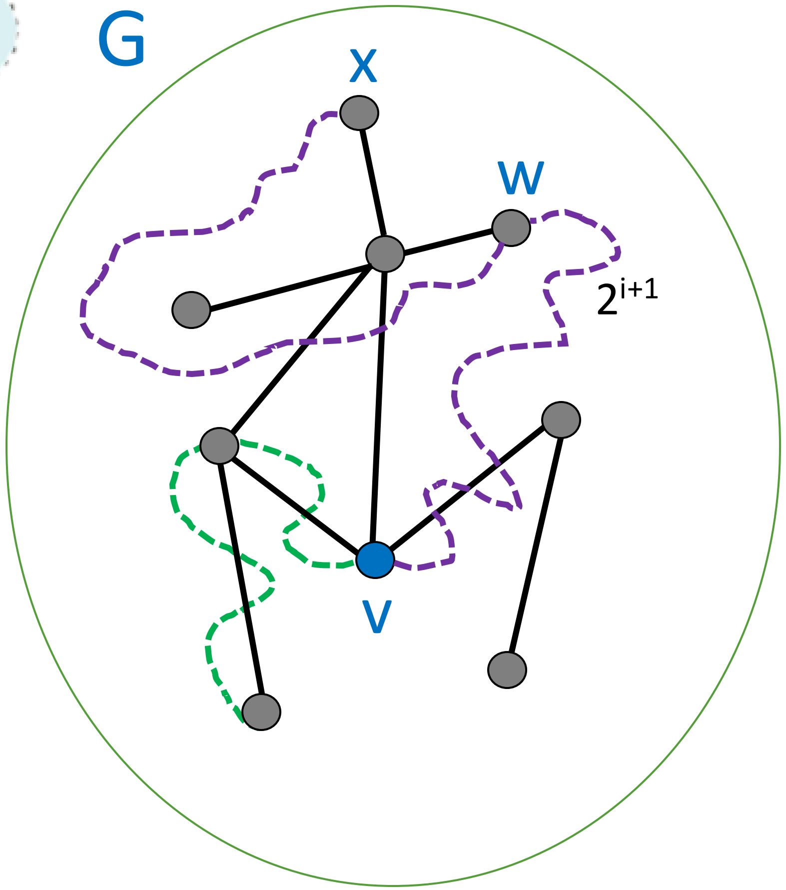
Track spare random walks. Use spare to **double** wanted ones.



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.

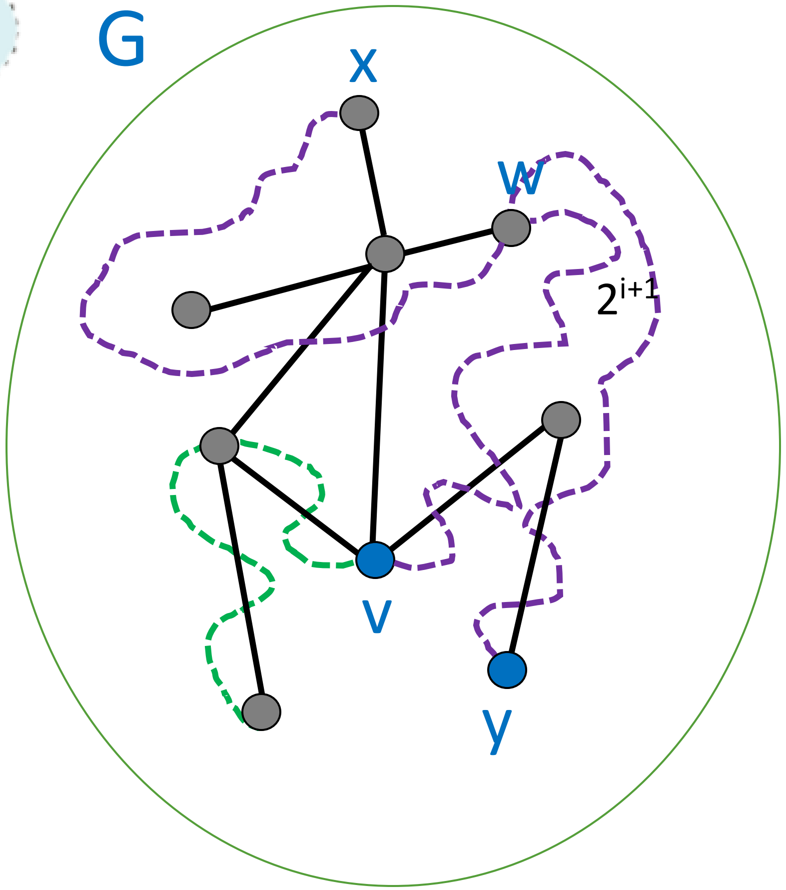


Random Walks: Doubling by Stitching

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?



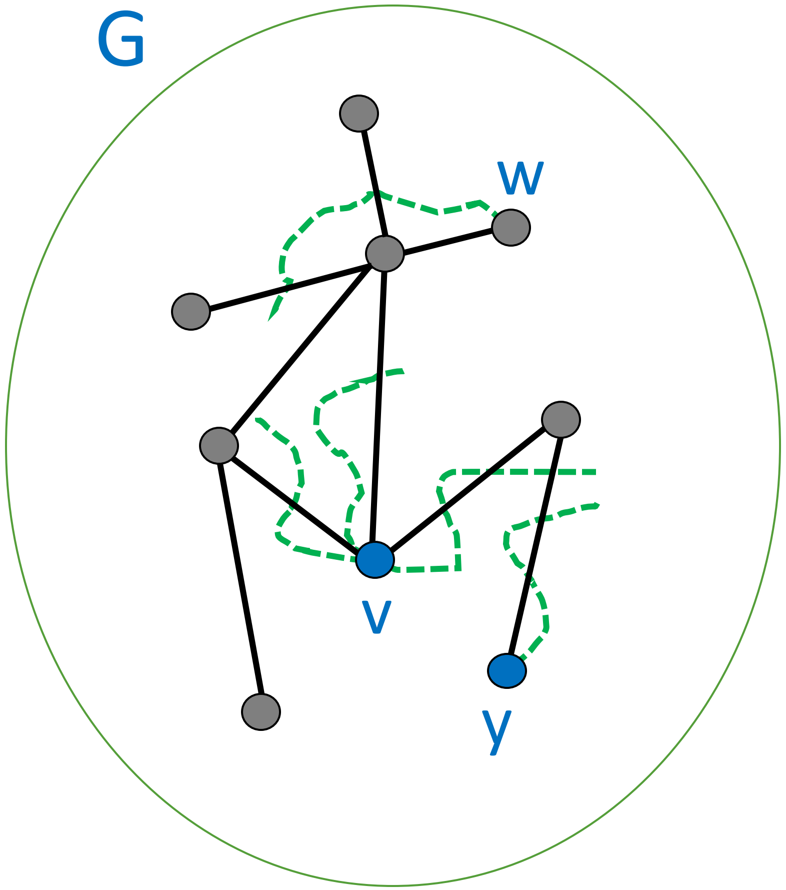
Random Walks: Follow Stationary Distribution

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

Random Walks: Follow Stationary Distribution

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

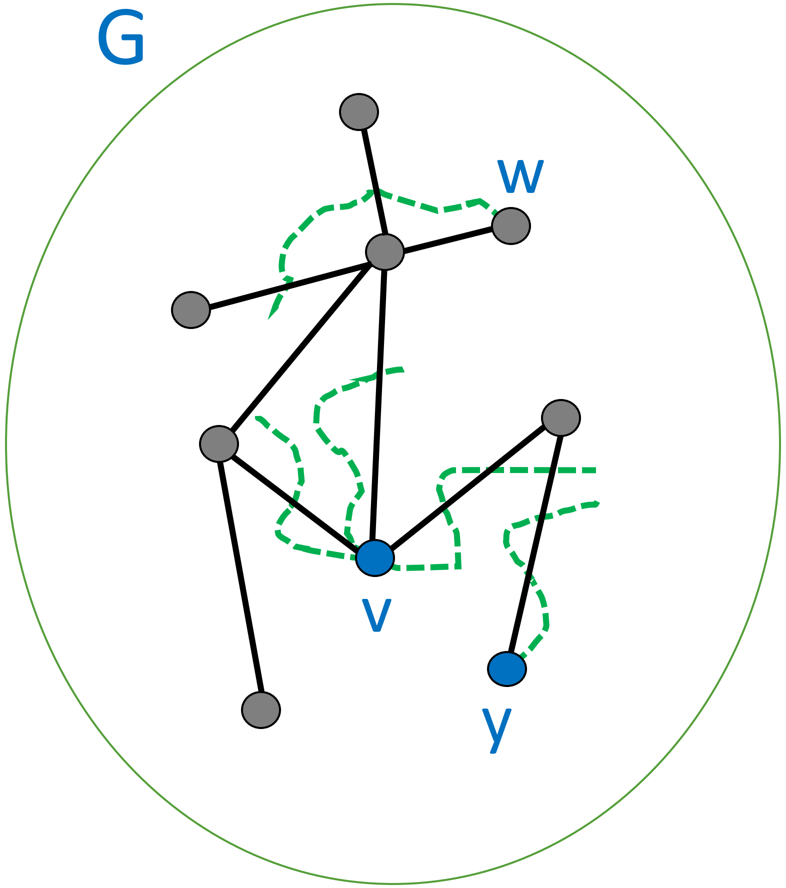
Each vertex v maintains proportionally to deg(v) random walks.



Random Walks: Follow Stationary Distribution

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

Each vertex v maintains proportionally to deg(v) random walks.



In **expectation**, after t steps there are proportionally to 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.

>=1/(2m)

PageRank for Directed Graphs

Input: Directed graph G^D

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

ε is the jumping probability

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

Space per machine: sublinear in n

Total space: $\tilde{O}((m + n^{1+o(1)})\, \varepsilon^{-4}\, \alpha^{-2})$.

(Prelude) Random Walks: Undirected vs Directed

Undirected graphs

Directed graphs

vs

(Prelude) Random Walks: Undirected vs Directed

Undirected graphs

Directed graphs

Stationary distribution is easy to compute: deg(v) / (2m).

VS

Stationary distribution of v is "nicely" lower-bounded.

(Prelude) Random Walks: Undirected vs Directed

Undirected graphs

Directed graphs

Stationary distribution is easy to compute: deg(v) / (2m).

Stationary distribution can be difficult to compute.

VS

Stationary distribution of v is "nicely" lower-bounded.

(Prelude) Random Walks: Undirected vs Directed

Undirected graphs

Directed graphs

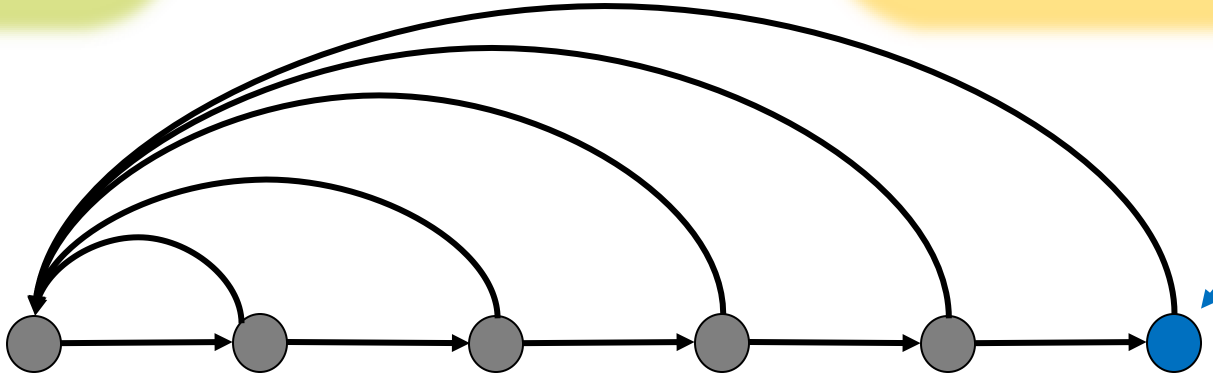
Stationary distribution is easy to compute: deg(v) / (2m).

Stationary distribution can be difficult to compute.

vs

Stationary distribution of v is "nicely" lower-bounded.

Stationary distribution of v can be $O(1/2^n)$.



PageRank: Undirected vs Directed Graphs

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

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

Output: Stationary distribution of T

PageRank: Undirected vs Directed Graphs

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

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

Output: Stationary distribution of T

Walk matrix of G.

Following P with prob. $1-\epsilon$.

Jumping to a random vertex

PageRank: Undirected vs Directed Graphs

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

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

Output: Stationary distribution of T

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

PageRank: Undirected vs Directed Graphs

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

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

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: $P = GD^{-1}$

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

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.

VS

PageRank: Undirected vs Directed Graphs

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

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

Output: Stationary distribution of T

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

Undirected graphs

T and P are "similar".

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

vs

Directed graphs

We do not know stationary distribution of T.

Stationary distribution of v w.r.t. T at least ε/n.


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.

Random walks for $(1\text{-}\delta)G+\delta\, G^D$.

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.

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

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

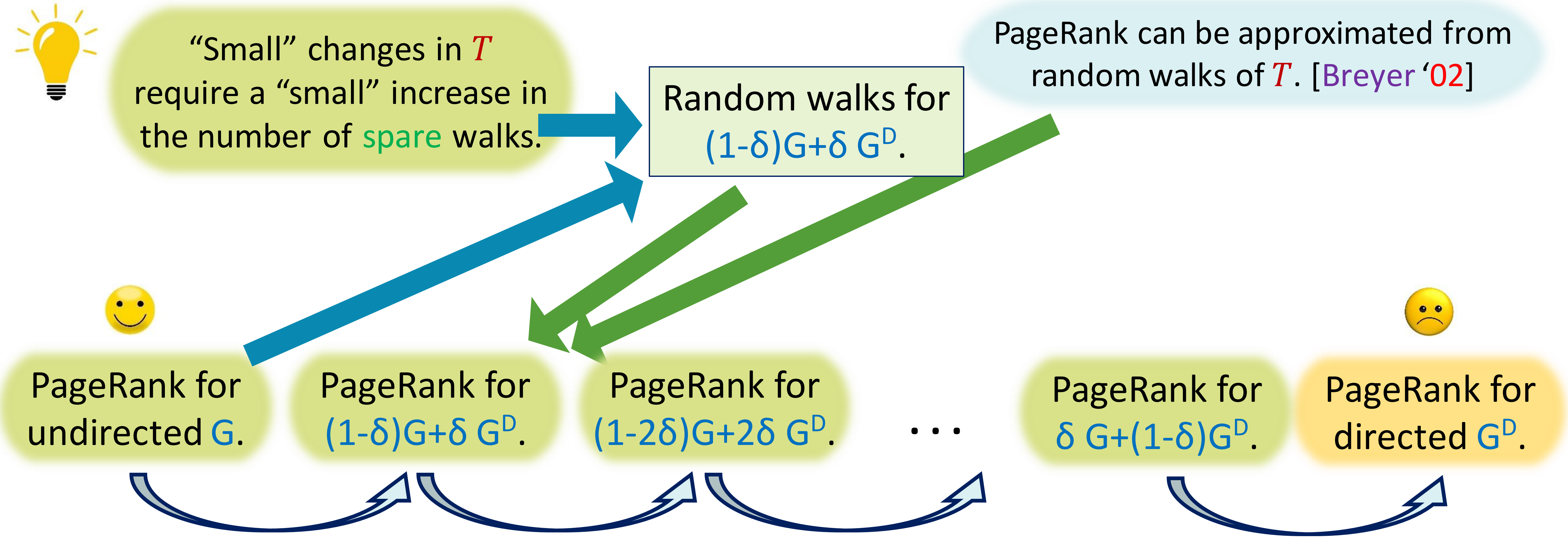
PageRank for undirected G.

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

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

PageRank for directed G^D.

PageRank: Molding Undirected to Directed



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.

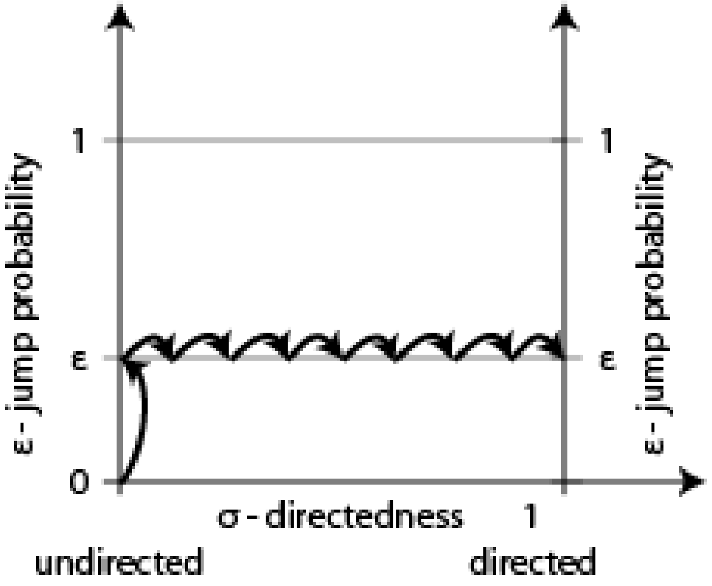
PageRank: Takeaway

1. The stationary distribution is lower-bounded by ε/n.

PageRank: Takeaway

1. The stationary distribution is lower-bounded by ε/n.

2. "Small" changes in a walk matrix affect the stationary distribution by little.

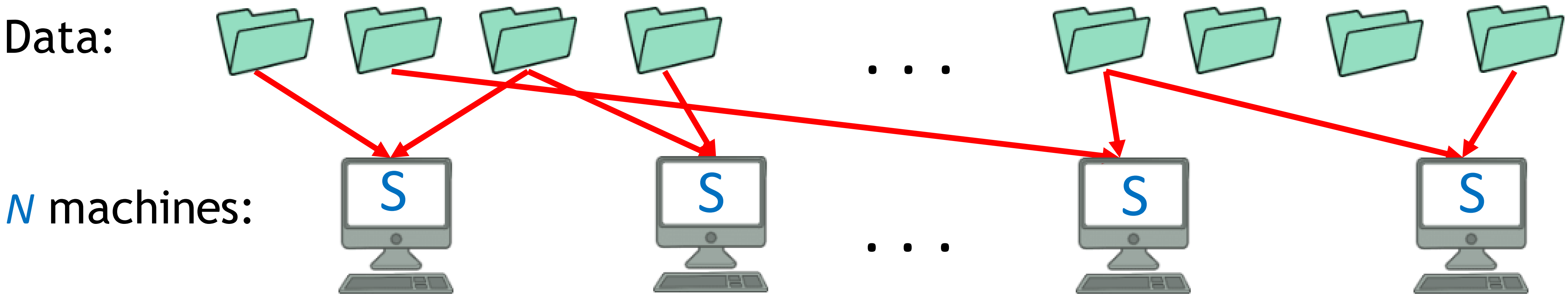


Massively Parallel Computation (MPC) round

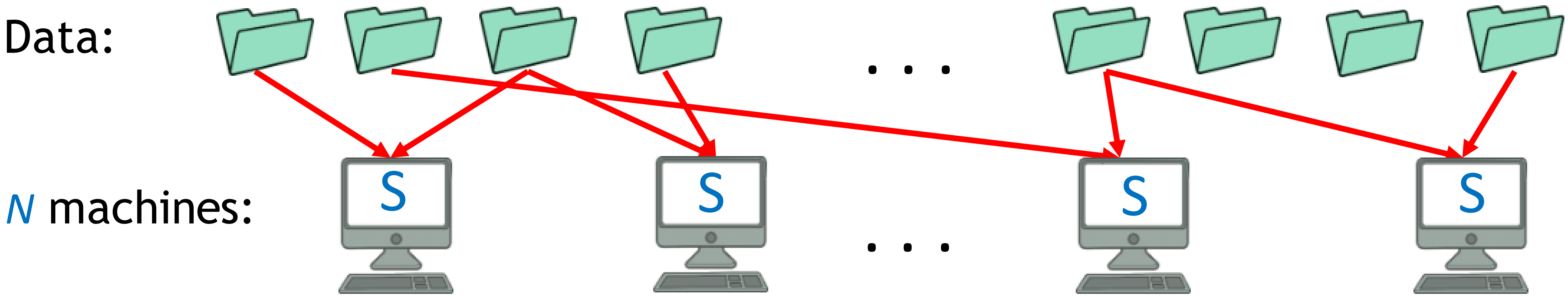
Data:

N machines:

Massively Parallel Computation (MPC) round

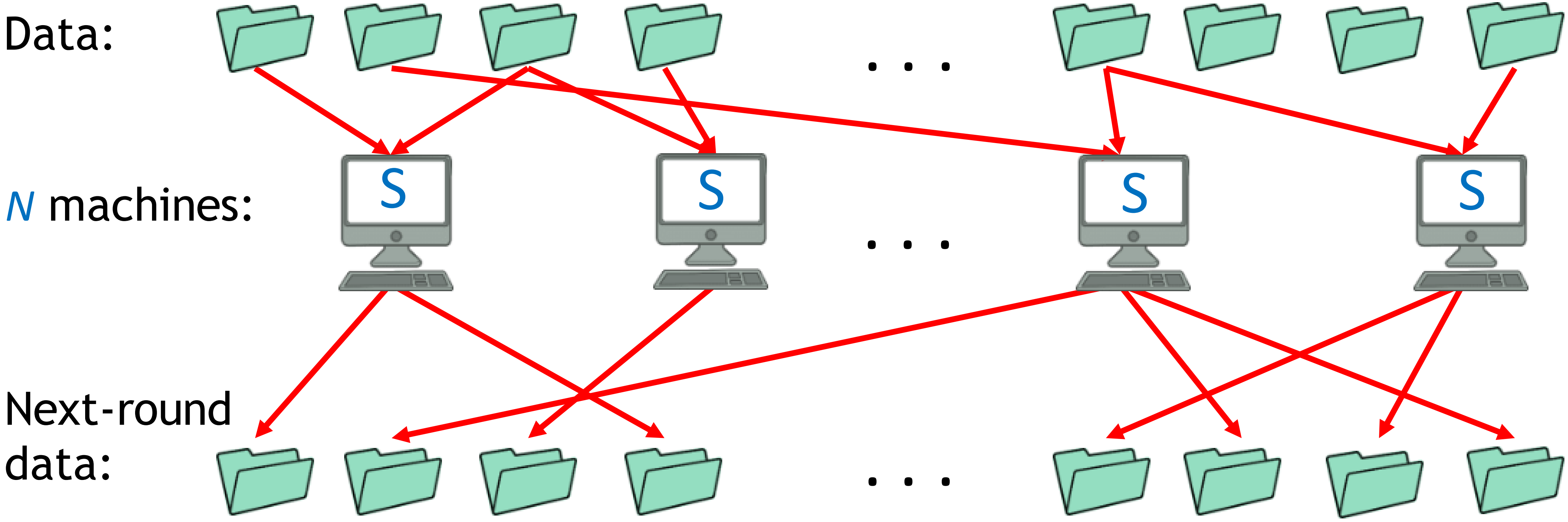


Massively Parallel Computation (MPC) round

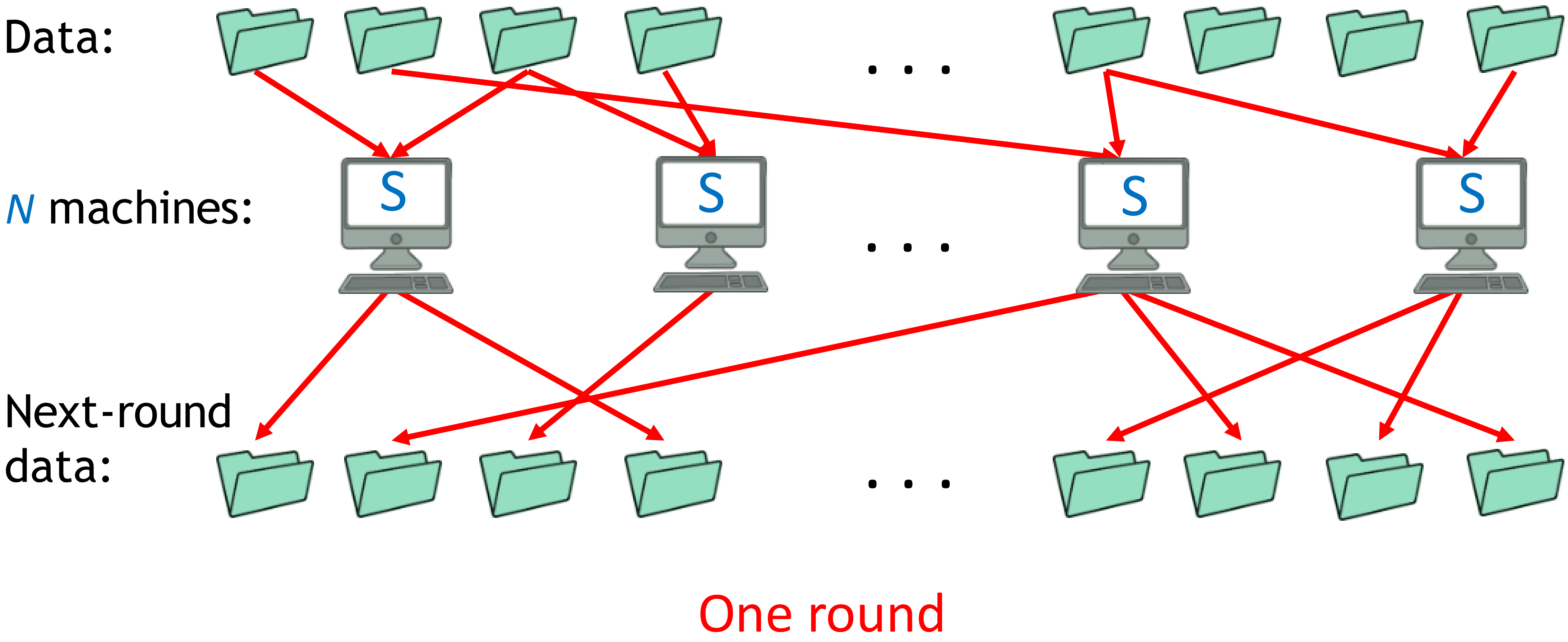


process data locally

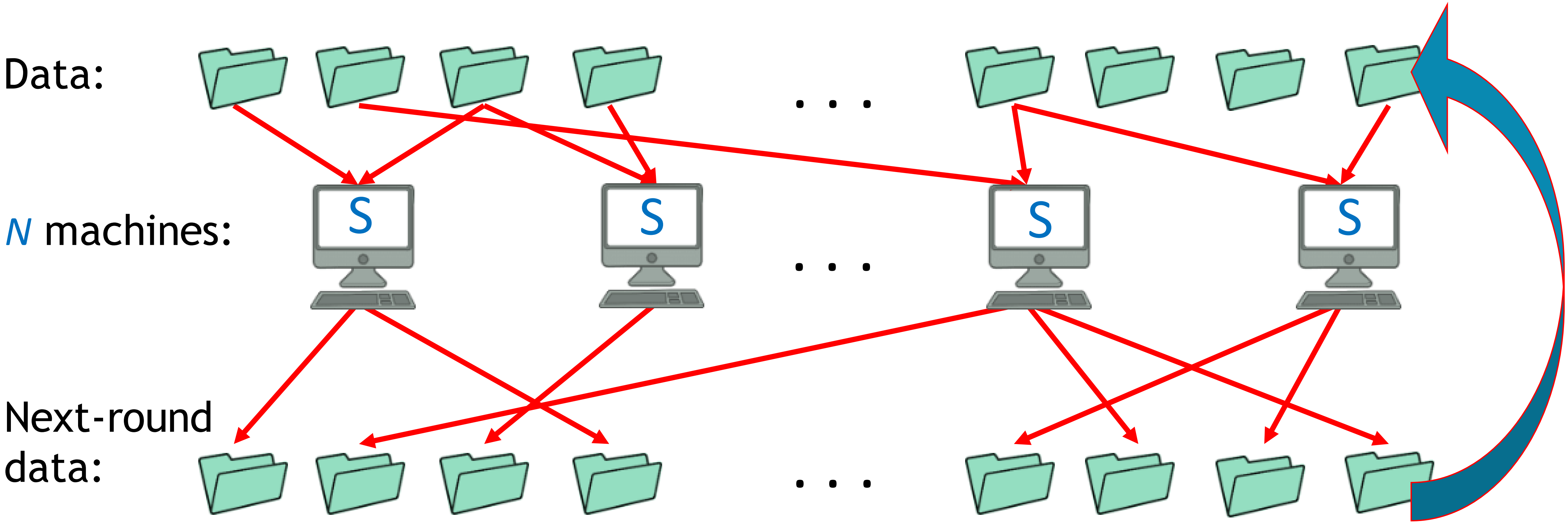
Massively Parallel Computation (MPC) round



Massively Parallel Computation (MPC) round



Massively Parallel Computation (MPC) round



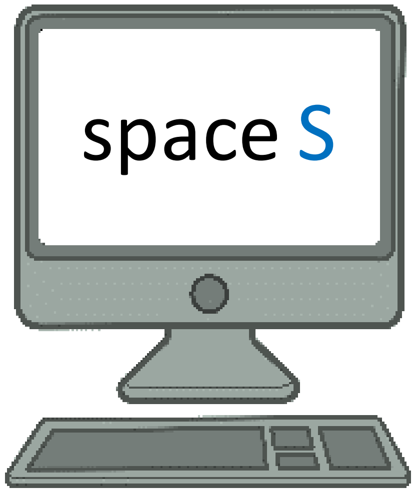
One round

Massively Parallel Computation (MPC) parameters

N = # of machines

S = space per machine

For graphs, $N * S = \Theta$(# of edges)

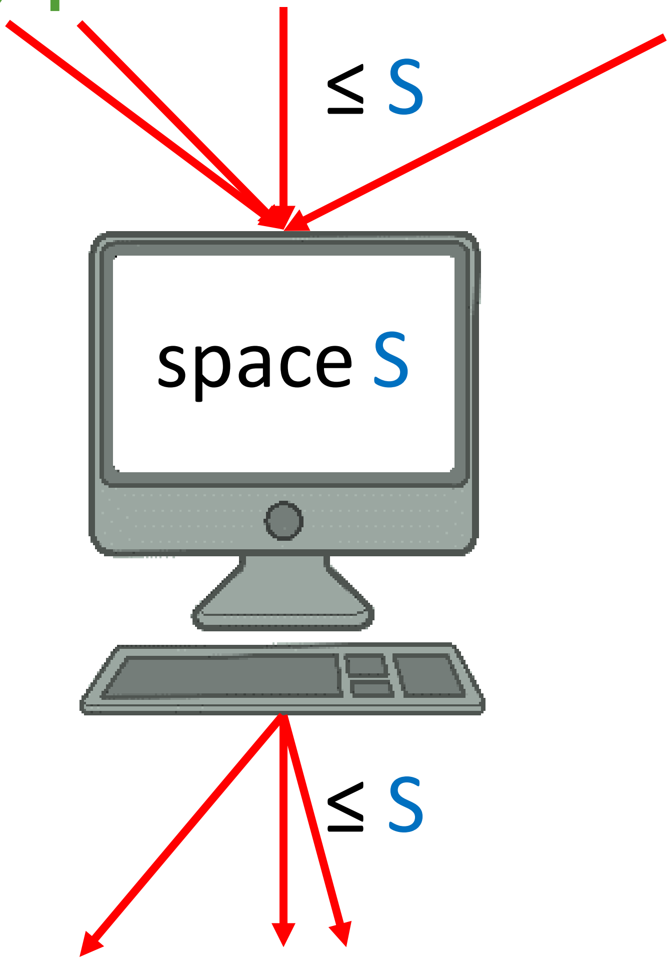


Massively Parallel Computation (MPC) parameters

N = # of machines

S = space per machine

For graphs, $N * S = \Theta(\text{\# of edges})$



<u>Interesting case</u>:

S much smaller than the input size

<u>Goal</u>:

make the small # of rounds