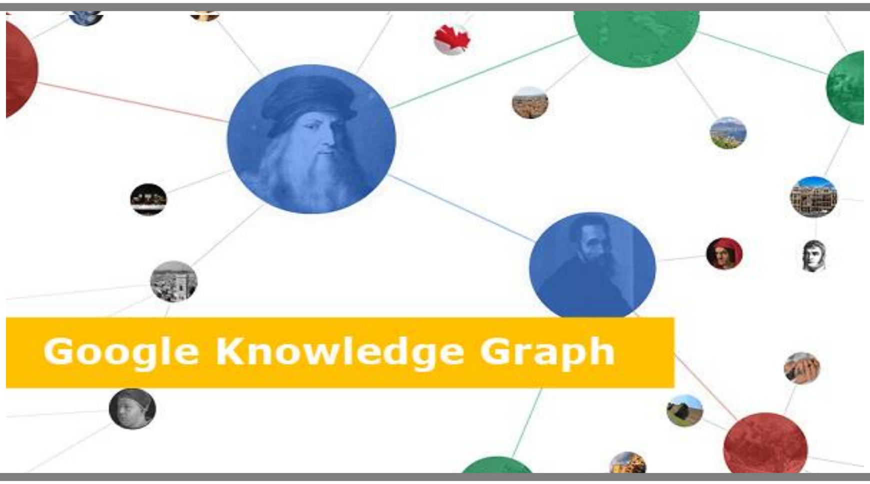
Graph algorithms in the Massively Parallel Computation (MPC) model

Slobodan Mitrović
(UC Davis)

Mandatory “Big Data” slides first …

amazon

12 million products
200 million Prime users



5 billion entities
500 billion facts

GPT-4

1+ trillion parameters

10.7 million square feet
(Inner Mongolia Information Park, China)
vs
4.2 million square feet
(Boeing Everett Factory, Washington, US)

hadoop Map Reduce

APACHE hadoop™

FLUME

gradoop

APACHE Spark™

Massively Parallel Computation (MPC) model

A theoretical abstraction of tools for handling massive data

Introduced:

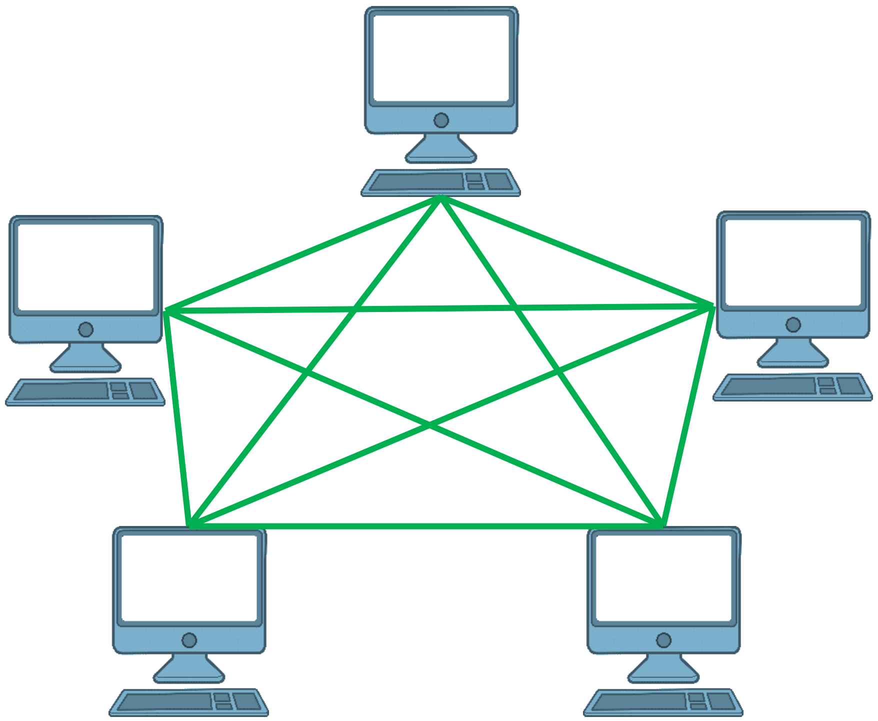
- [Dean, Ghemawat, ‘04, ‘08]
- [Karloff, Suri, Vassilvitskii, ‘10]
- [Goodrich, Sitchinava, Zhang, ‘11]

Examples:

- MapReduce [Dean, Ghemawat, ‘04, ‘08]
- Hadoop [White, ‘12]
- Pregel [Google, ‘09]
- Dryad [Isard, Budiu, Yu, Birrell, Fetterly, ‘07]
- Spark [Zaharia, Chowdhury, Franklin, Shenker, Stoica, ‘10]

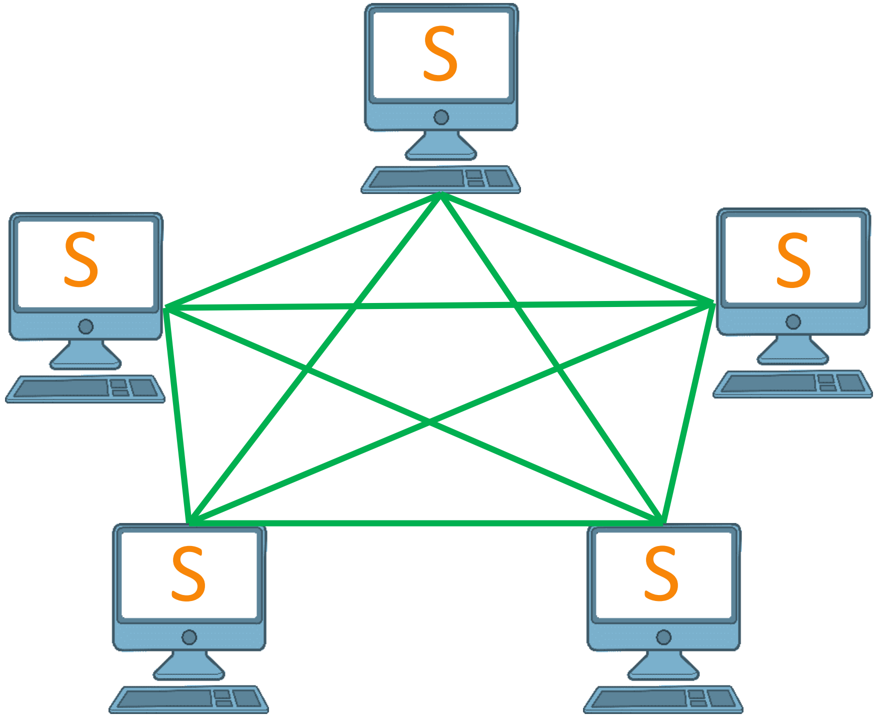
Massively Parallel Computation (MPC)

All-to-all synchronous-round communication



Massively Parallel Computation (MPC)

All-to-all synchronous-round communication

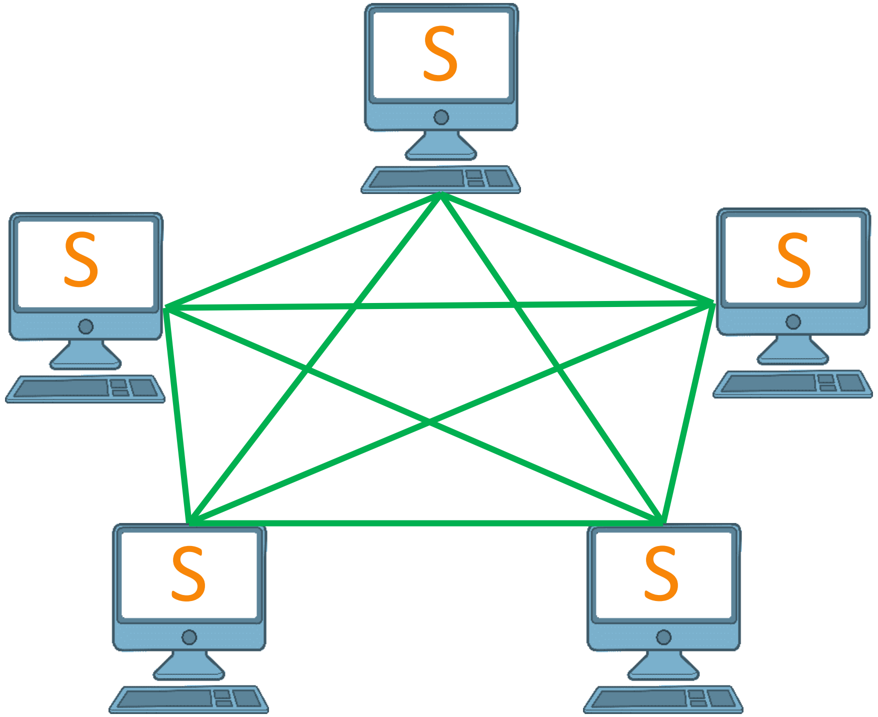


Parametrized:

- *T* machines
- Space *S* per machine (RAM)
- (desired) $T * S \approx$ input size

Massively Parallel Computation (MPC)

All-to-all synchronous-round communication



Parametrized:

T machines

Space S per machine (RAM)

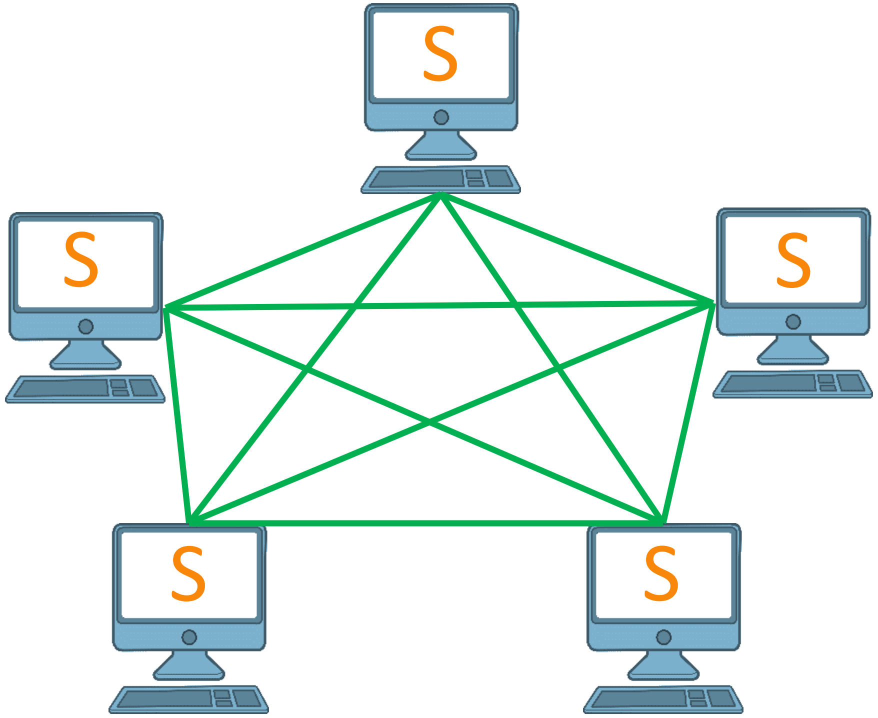
(desired) $T * S \approx$ input size

Constraints per round:

Machine receives/sends at most S bits

Massively Parallel Computation (MPC)

All-to-all synchronous-round communication



Parametrized:

T machines

Space S per machine (RAM)

(desired) $T * S \approx$ input size

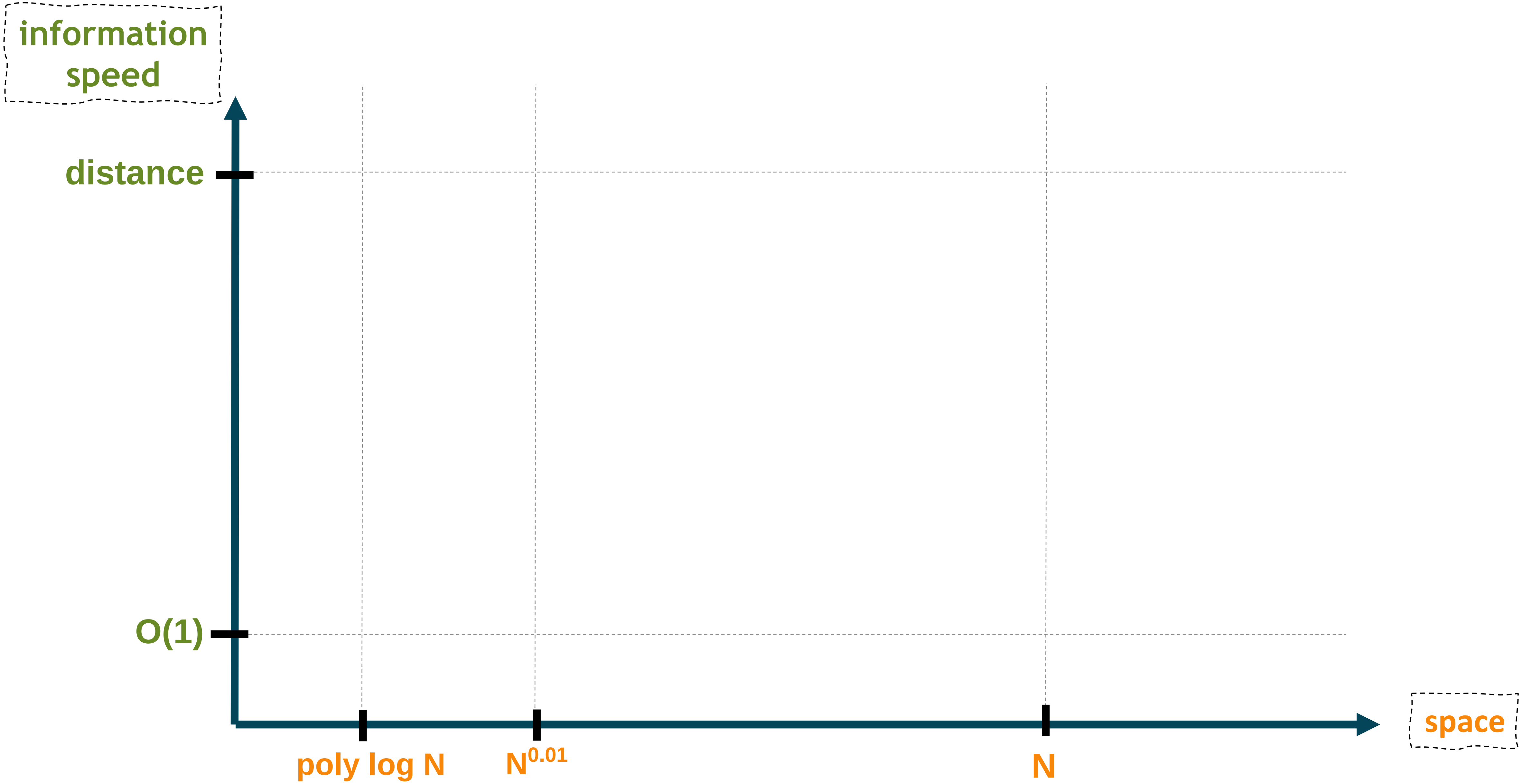
Constraints per round:

Machine receives/sends at most S bits

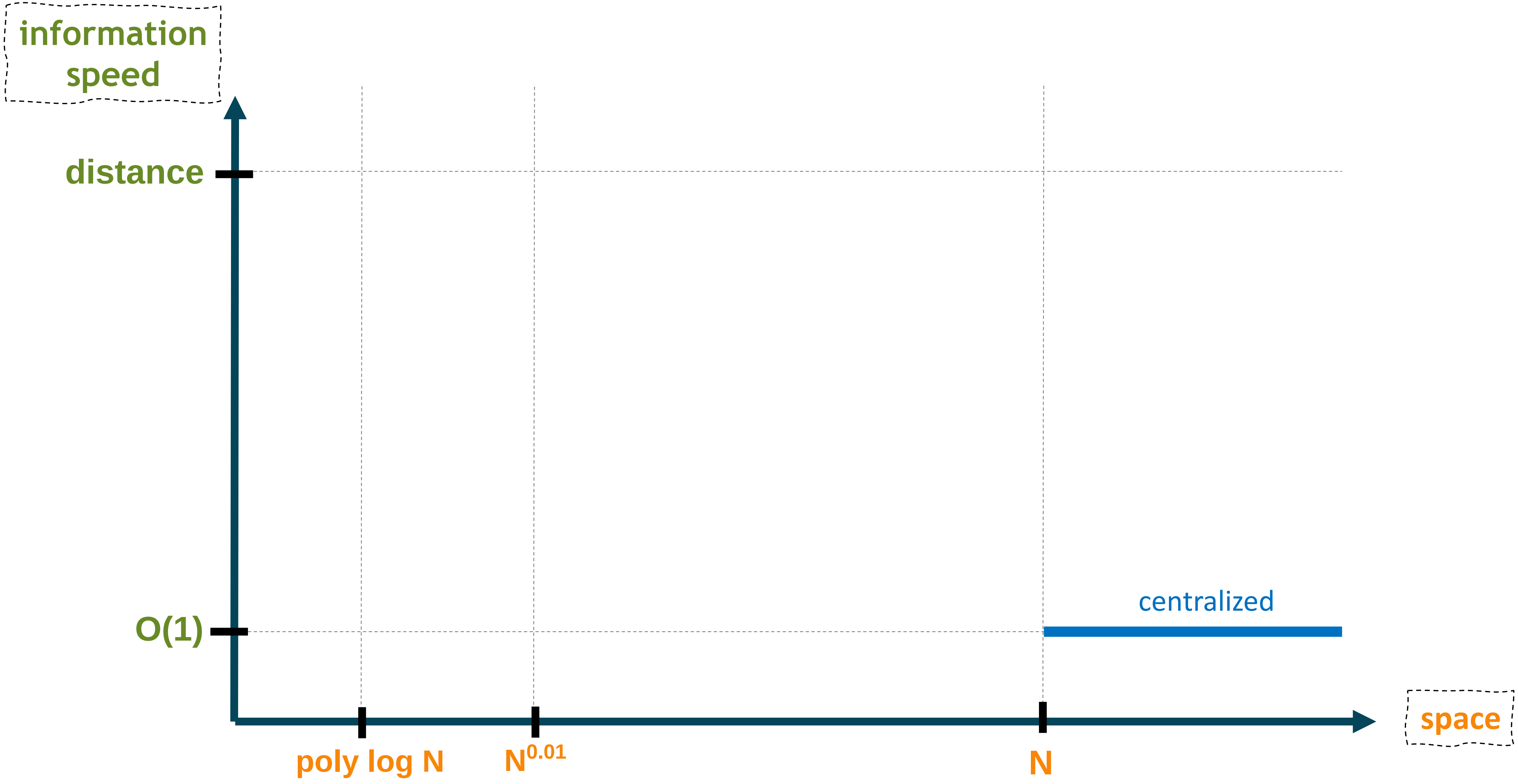
Goal:

As few rounds as possible.

N = input size

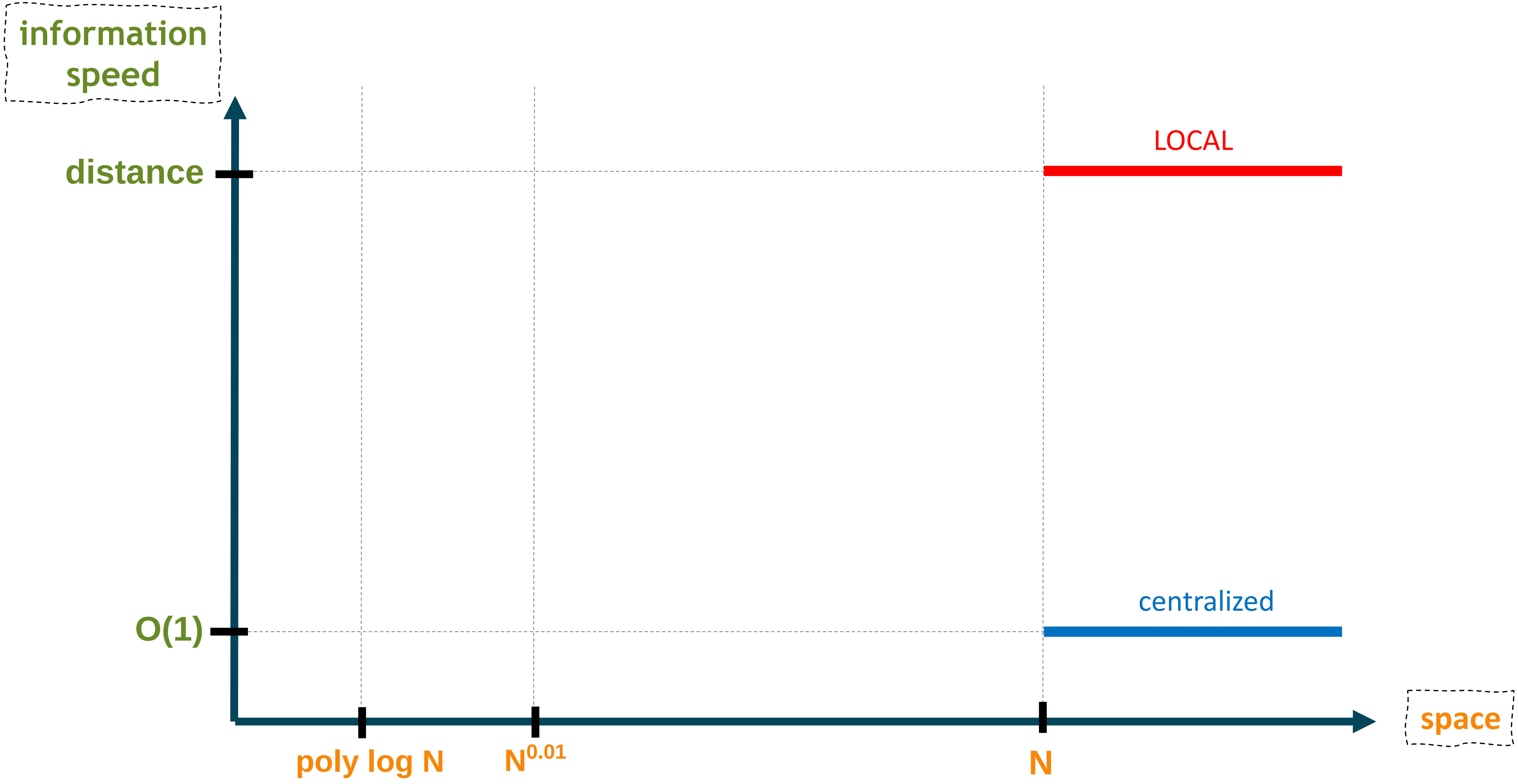


N = input size

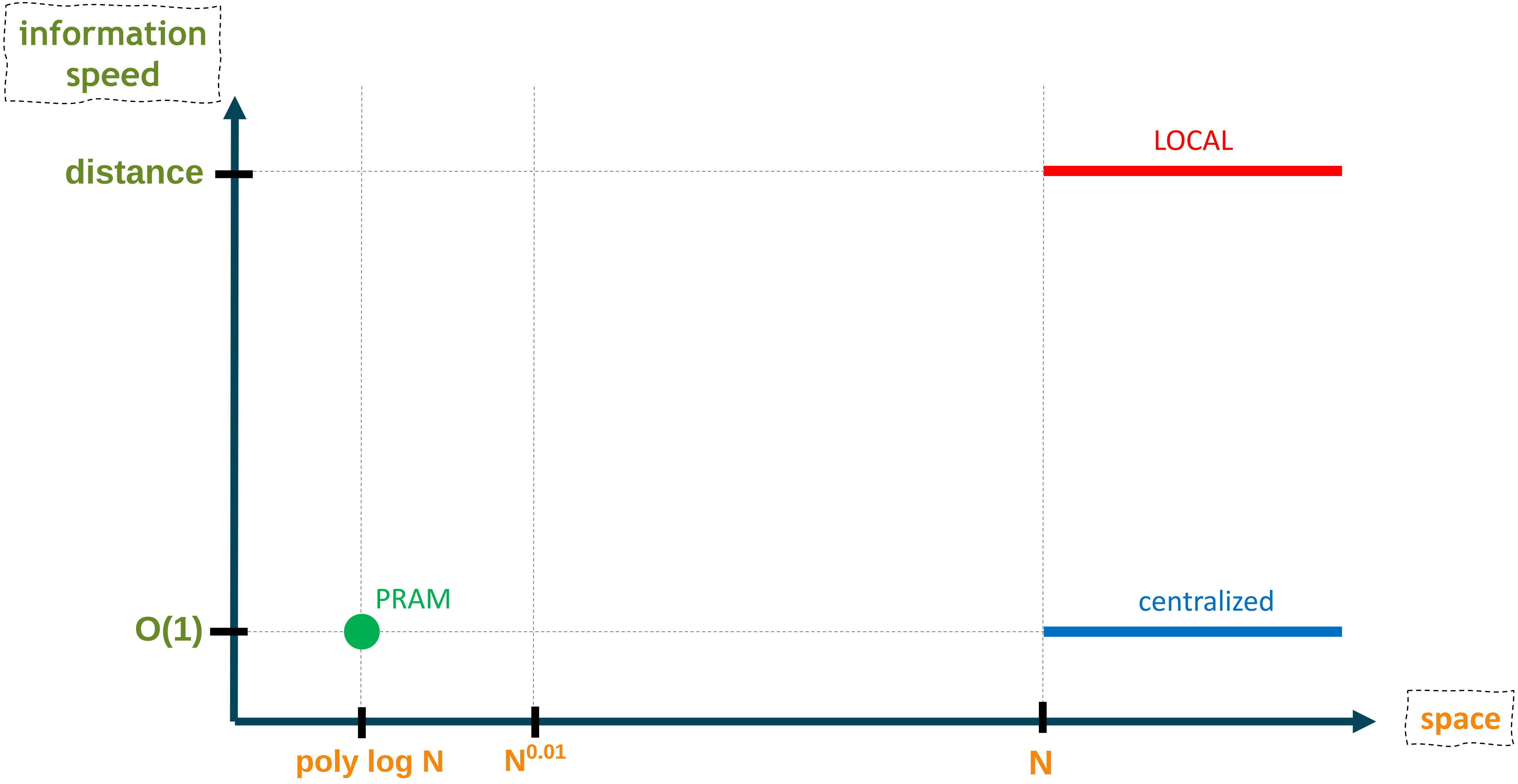


information speed
distance
O(1)
centralized
space
poly log N
N^0.01
N

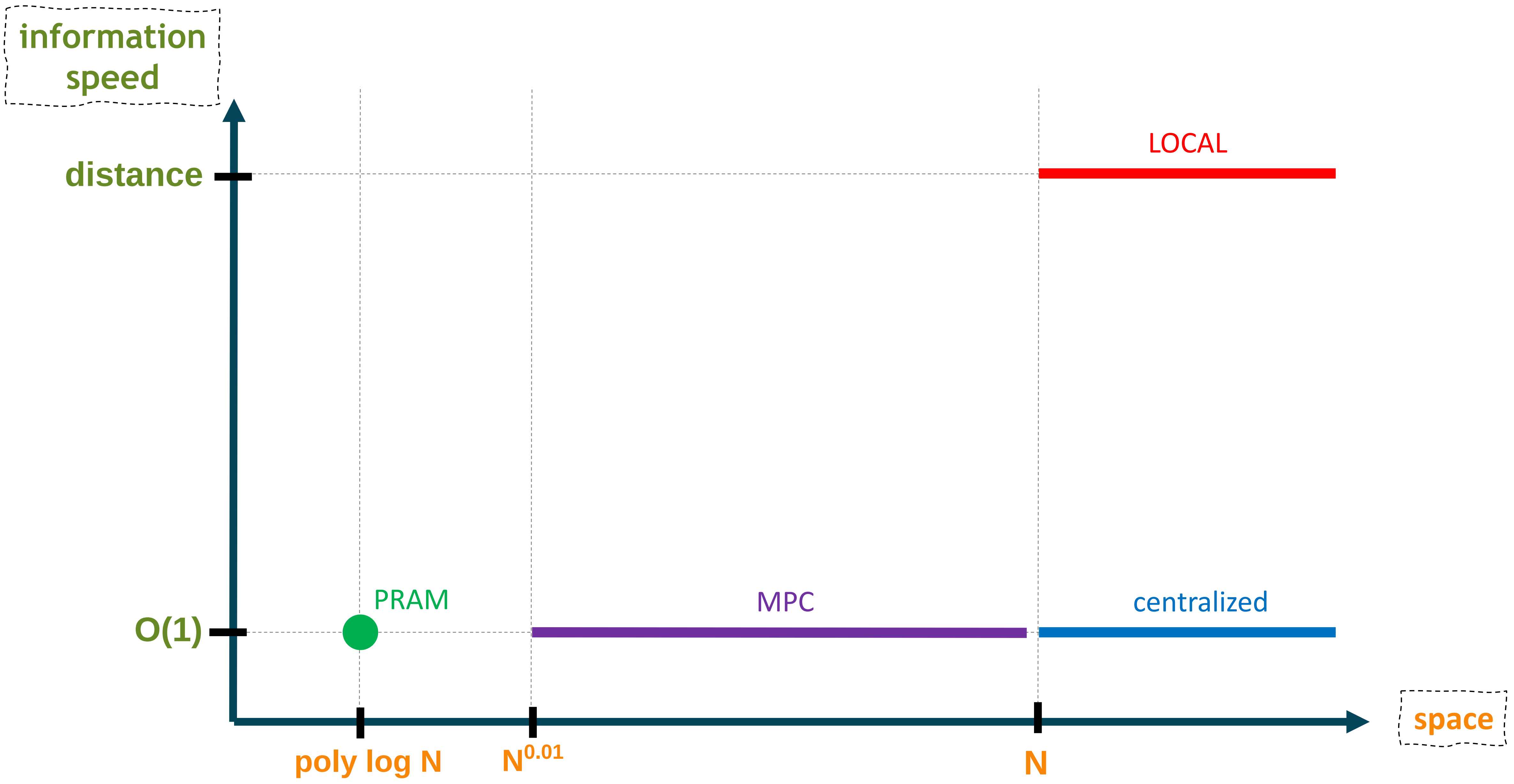
N = input size


information speed
distance
O(1)
LOCAL
centralized
poly log N
$N^{0.01}$
N
space

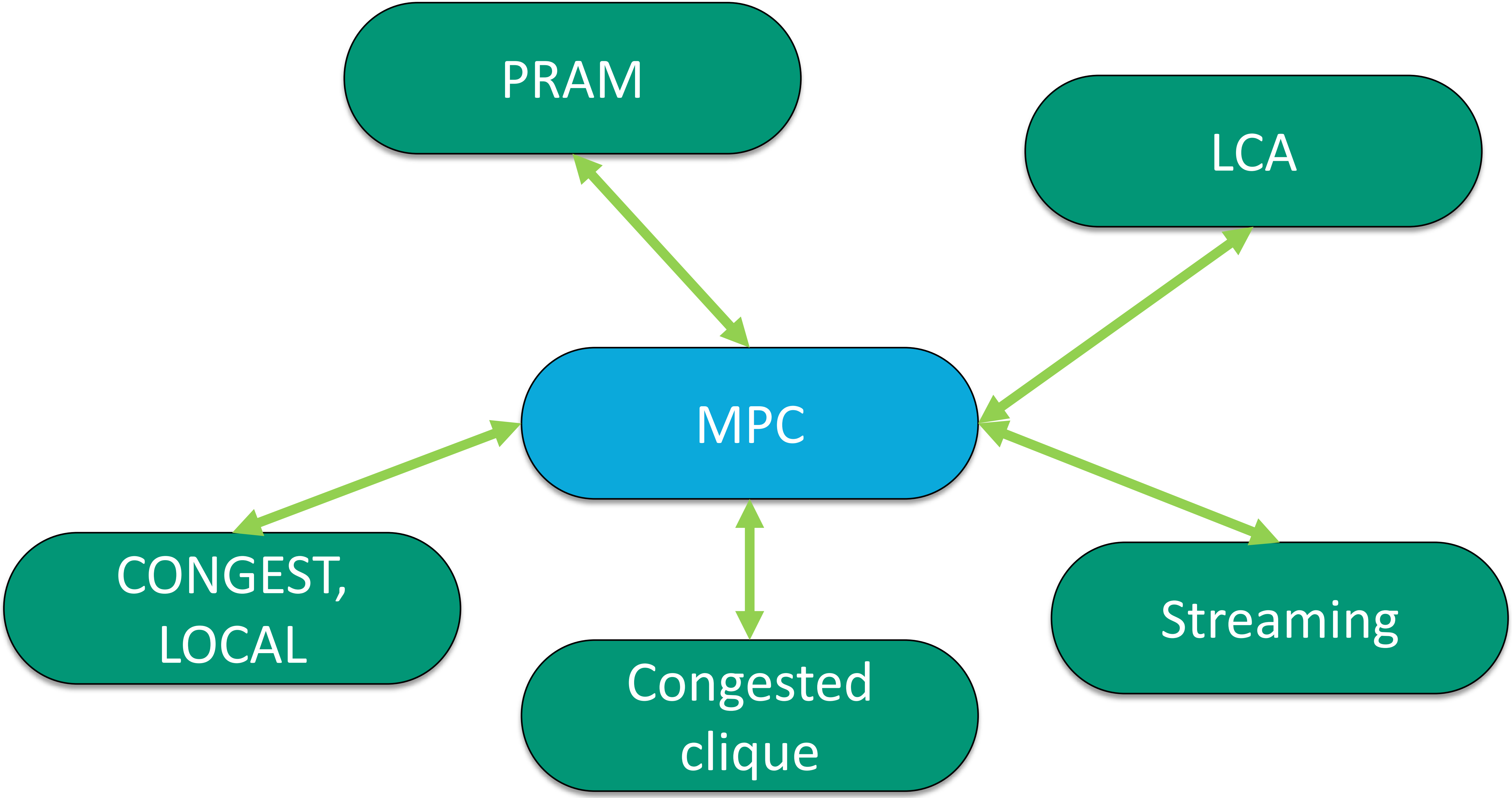
N = input size


information speed
distance
O(1)
LOCAL
PRAM
centralized
poly log N
N0.01
N
space

N = input size



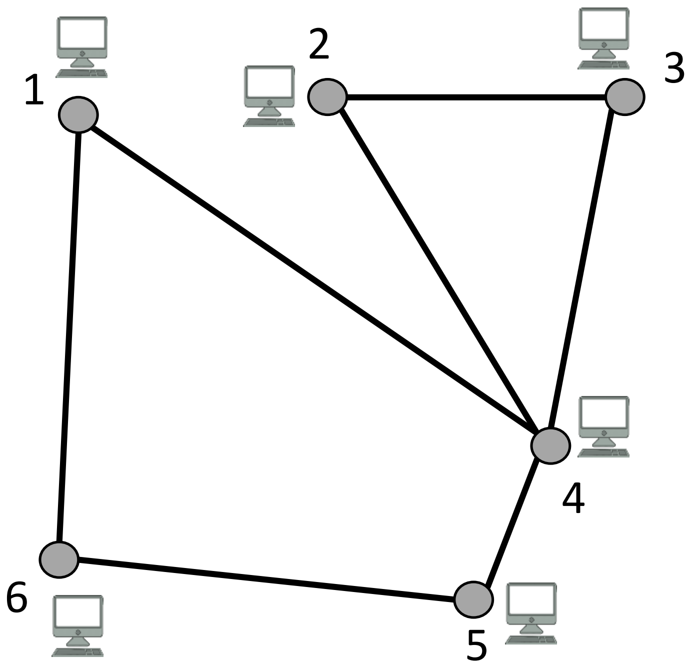
information speed
distance
O(1)
LOCAL
PRAM
MPC
centralized
poly log N
N0.01
N
space


PRAM
LCA
MPC
CONGEST,
LOCAL
Congested
clique
Streaming

Today: A single technique on a specific problem.

Simulation via Round Compression

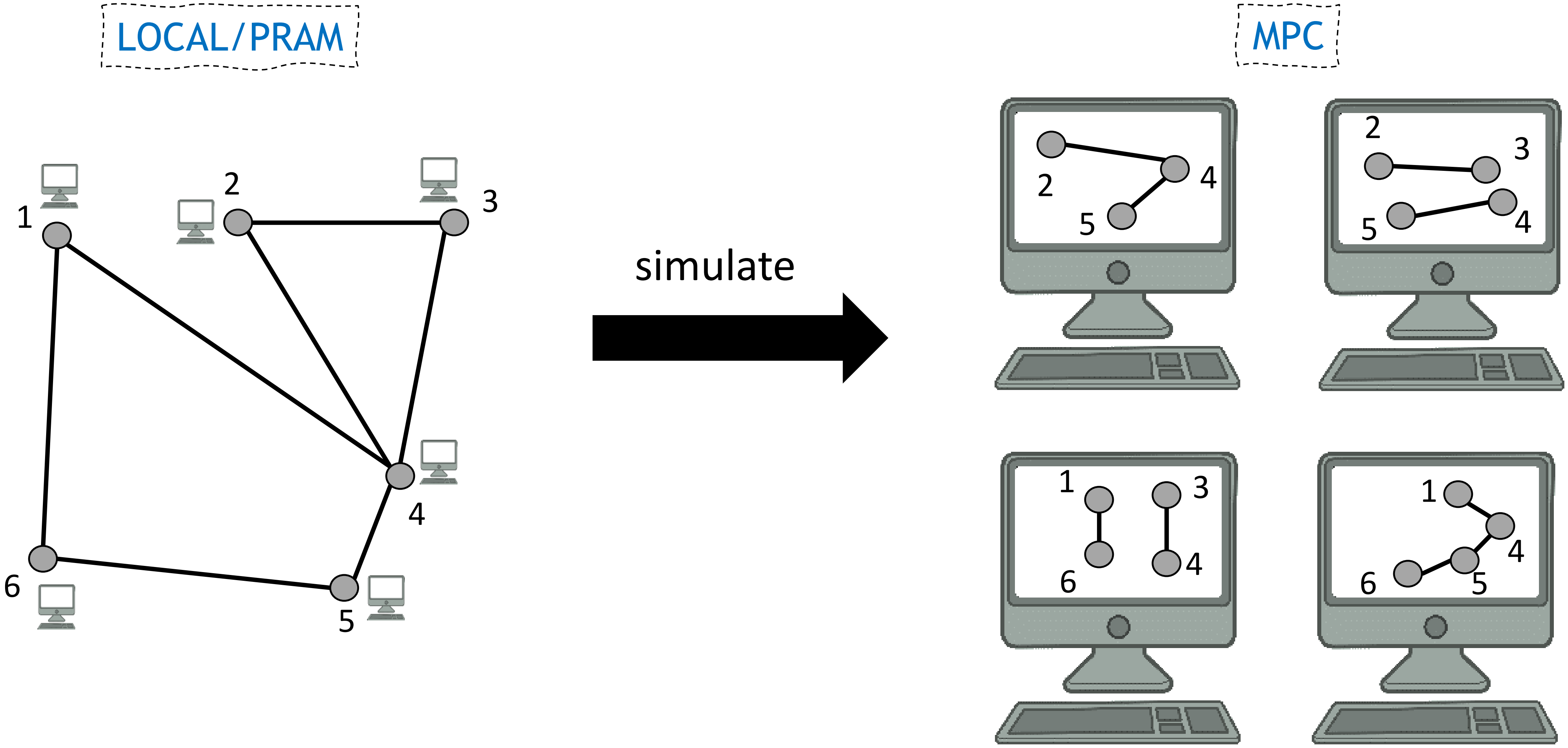
LOCAL/PRAM



Algorithm: A

Rounds: T

Simulation via Round Compression



Algorithm: A   Algorithm: ≈A

Rounds: T   Rounds: o(T)

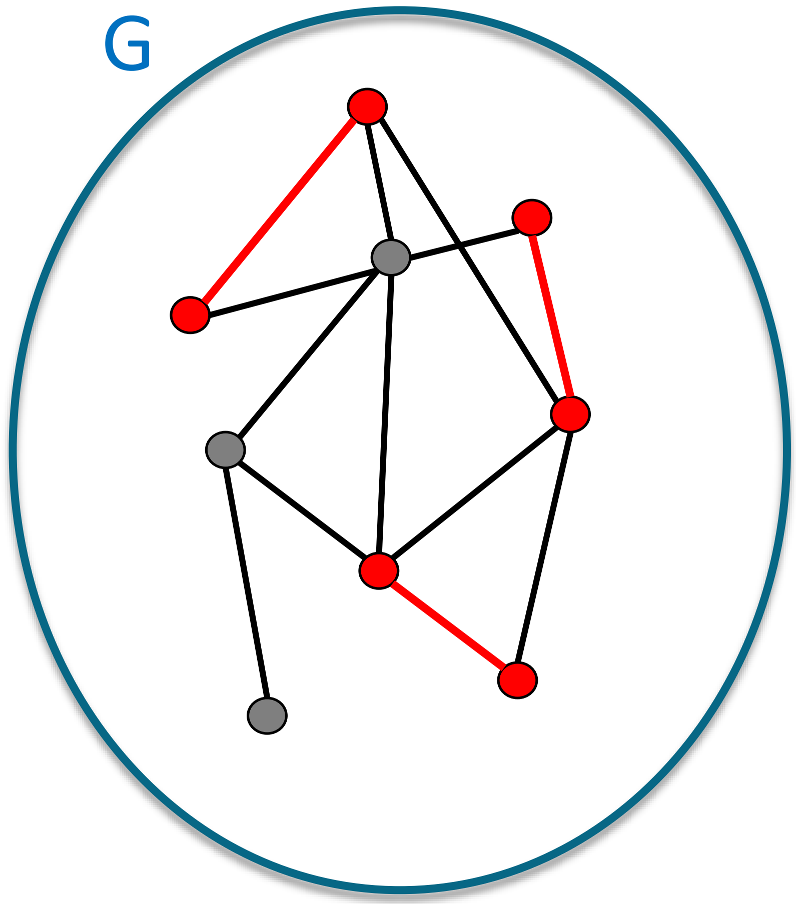
Approximate Maximum Matching in MPC with O(n) space per machine

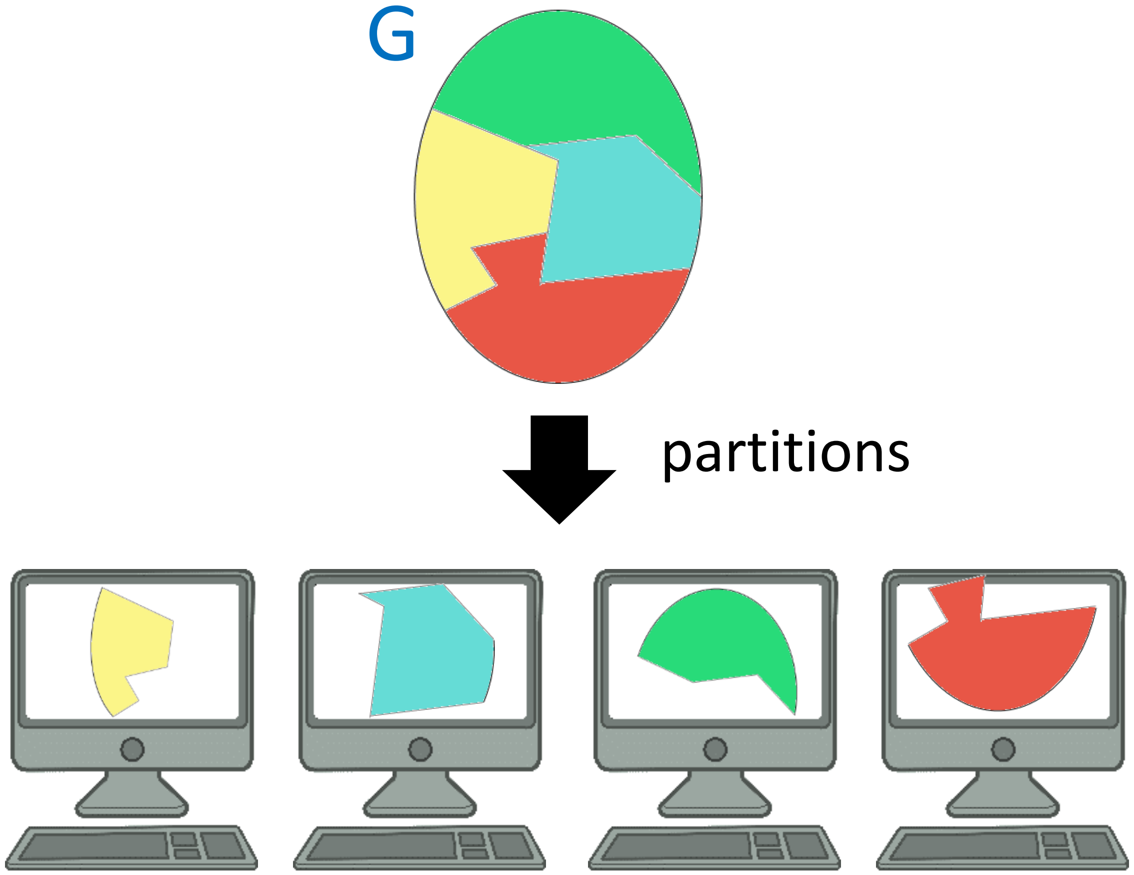
Input:

- an unweighted graph $G = (V, E)$

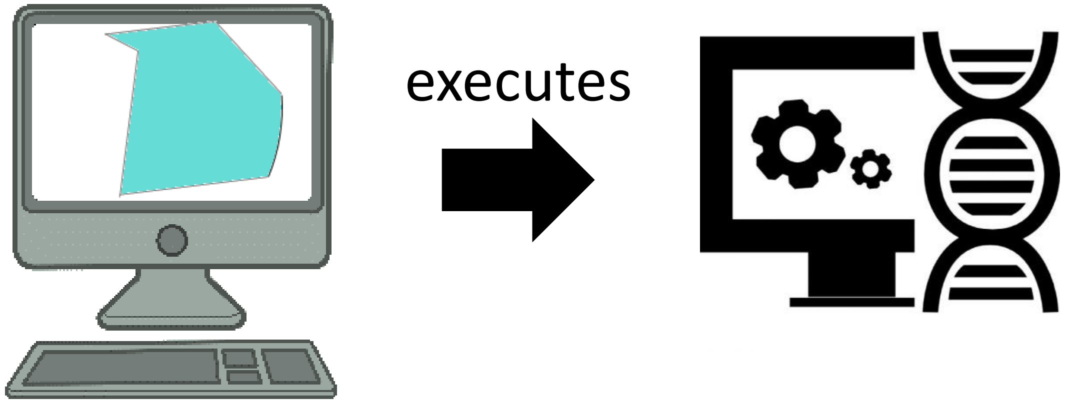
Output:

- a constant-factor approximate *maximum matching*

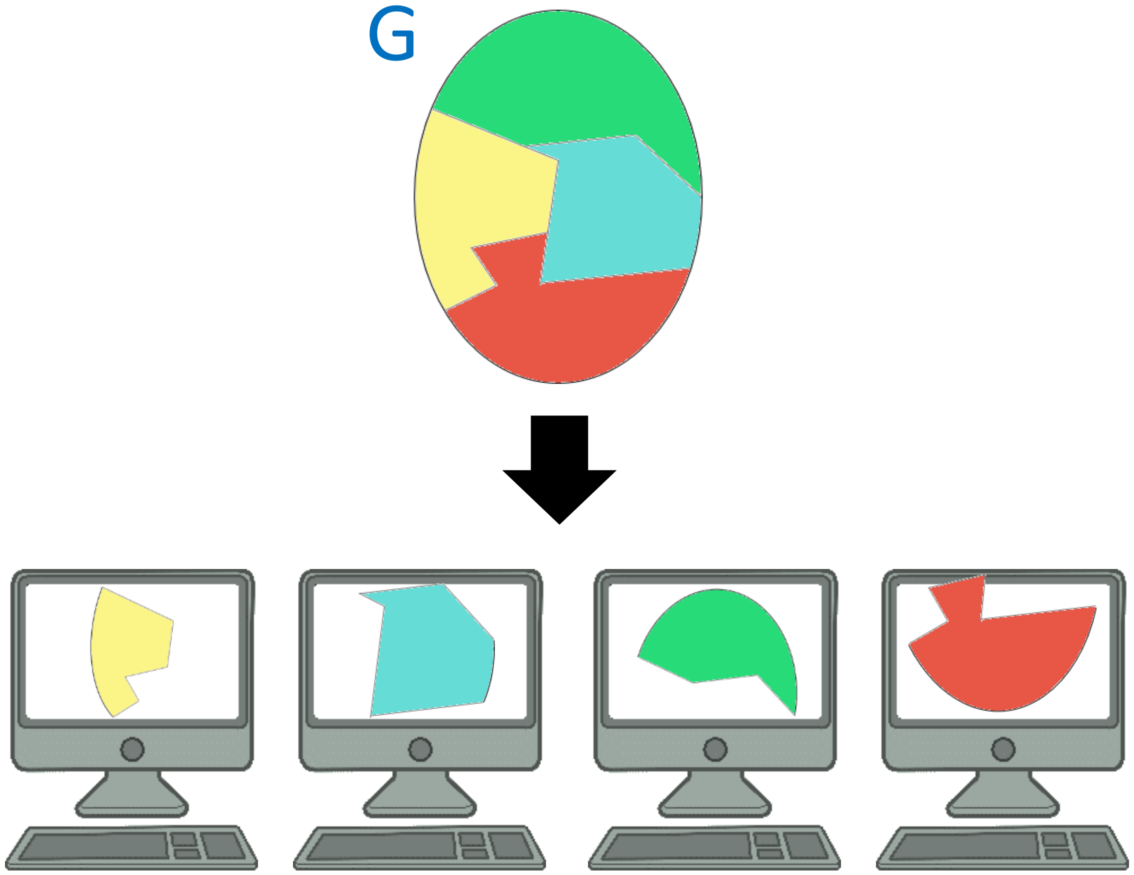




How to partition the graph?



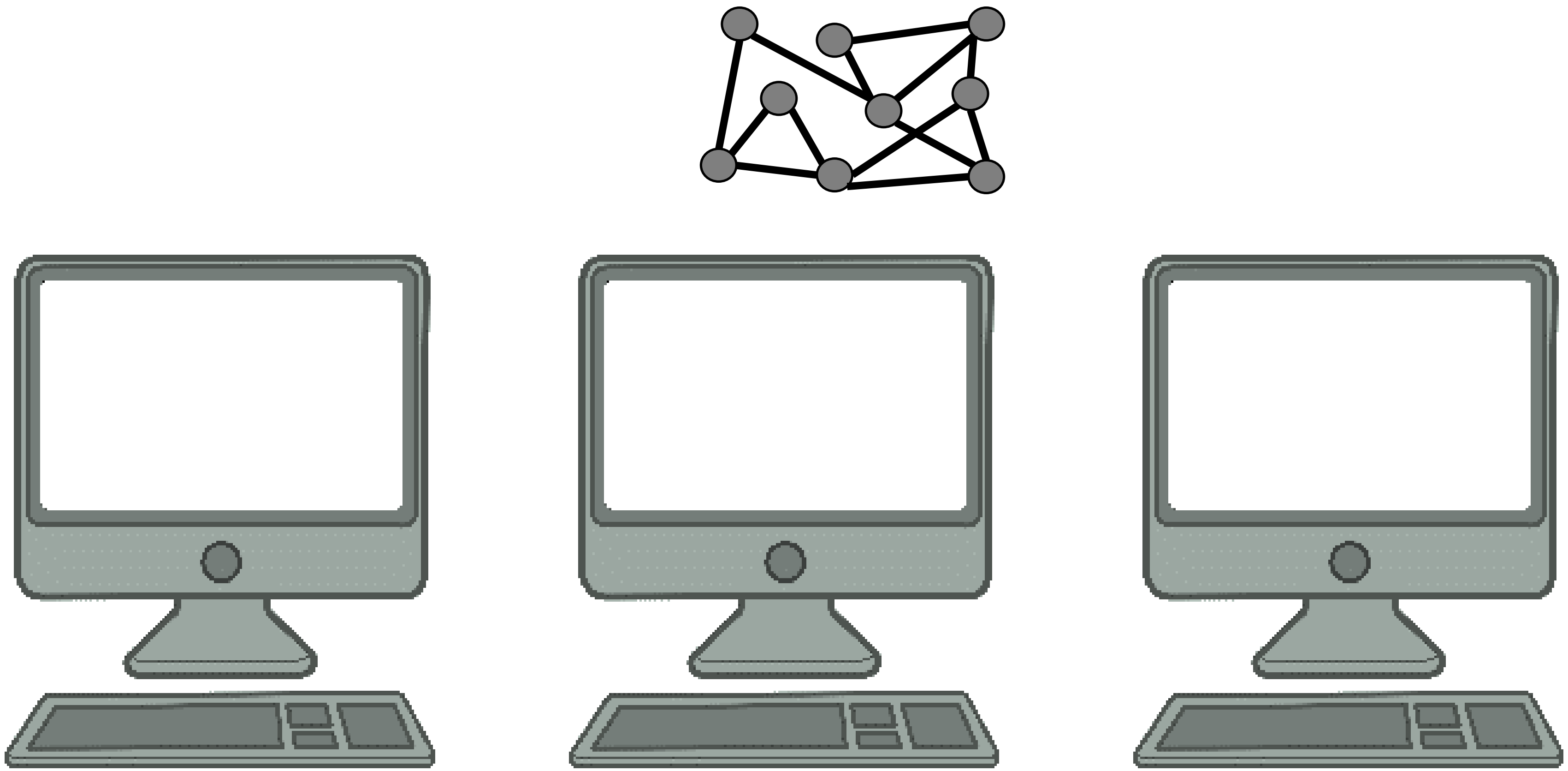
What local algorithm to use?



Random vertex partitioning

- [Czumaj, Łącki, Mądry, Mitrović, Onak, Sankowski ’17]
- [Ghaffari, Gouleakis, Konrad, Mitrović, Rubinfeld ’18]
- [Assadi, Bateni, Bernstein, Mirrokni, Stein ’19]
- [Behnezhad, Hajiaghayi, Harris ’19]
- [Ghaffari, Lattanzi, Mitrović ’19]
- [Biswas, Eden, Liu, Mitrović, Rubinfeld ’22]

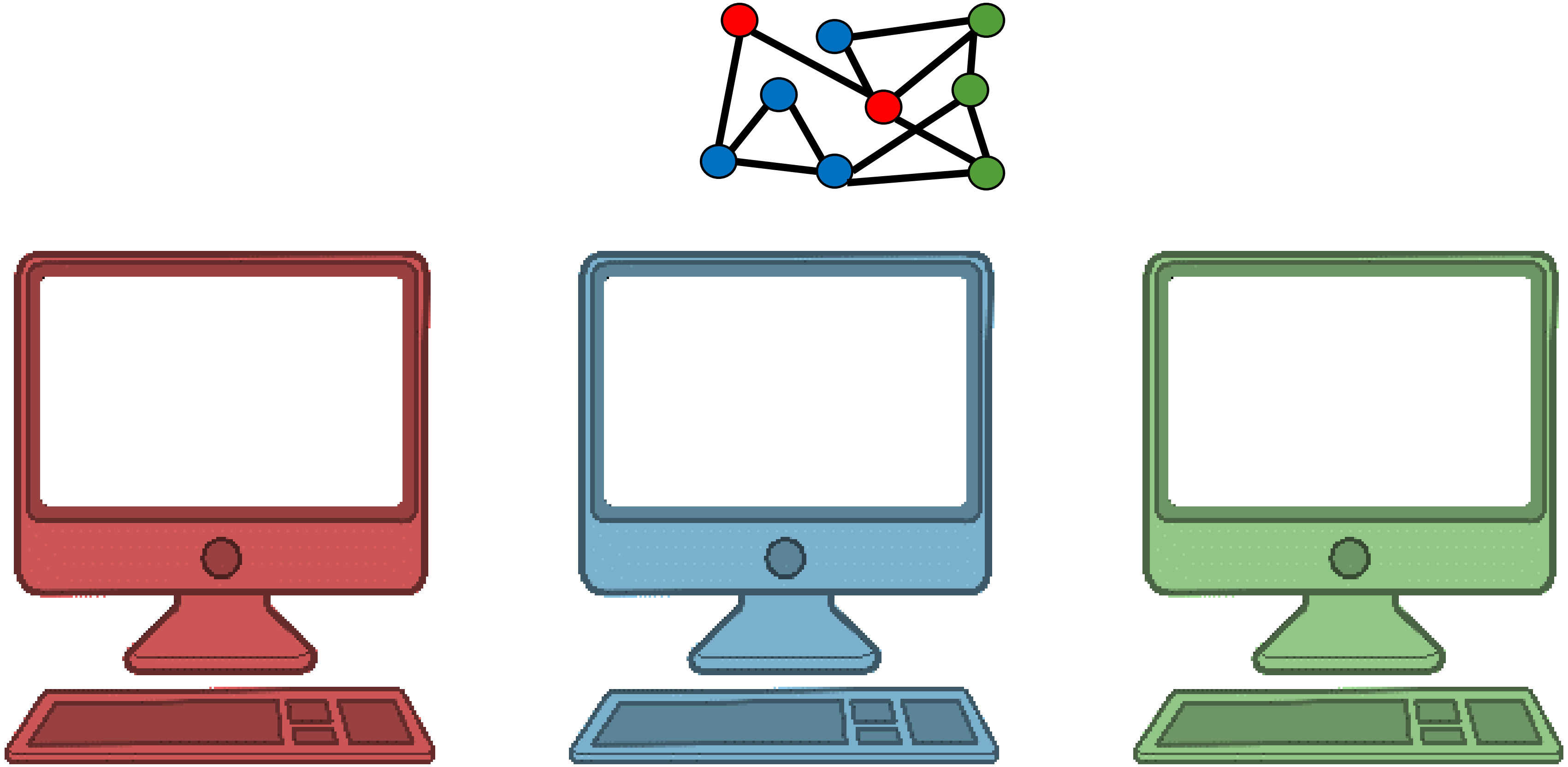
Random vertex partitioning



Random vertex partitioning

$\sqrt{\Delta}$ colors/machines

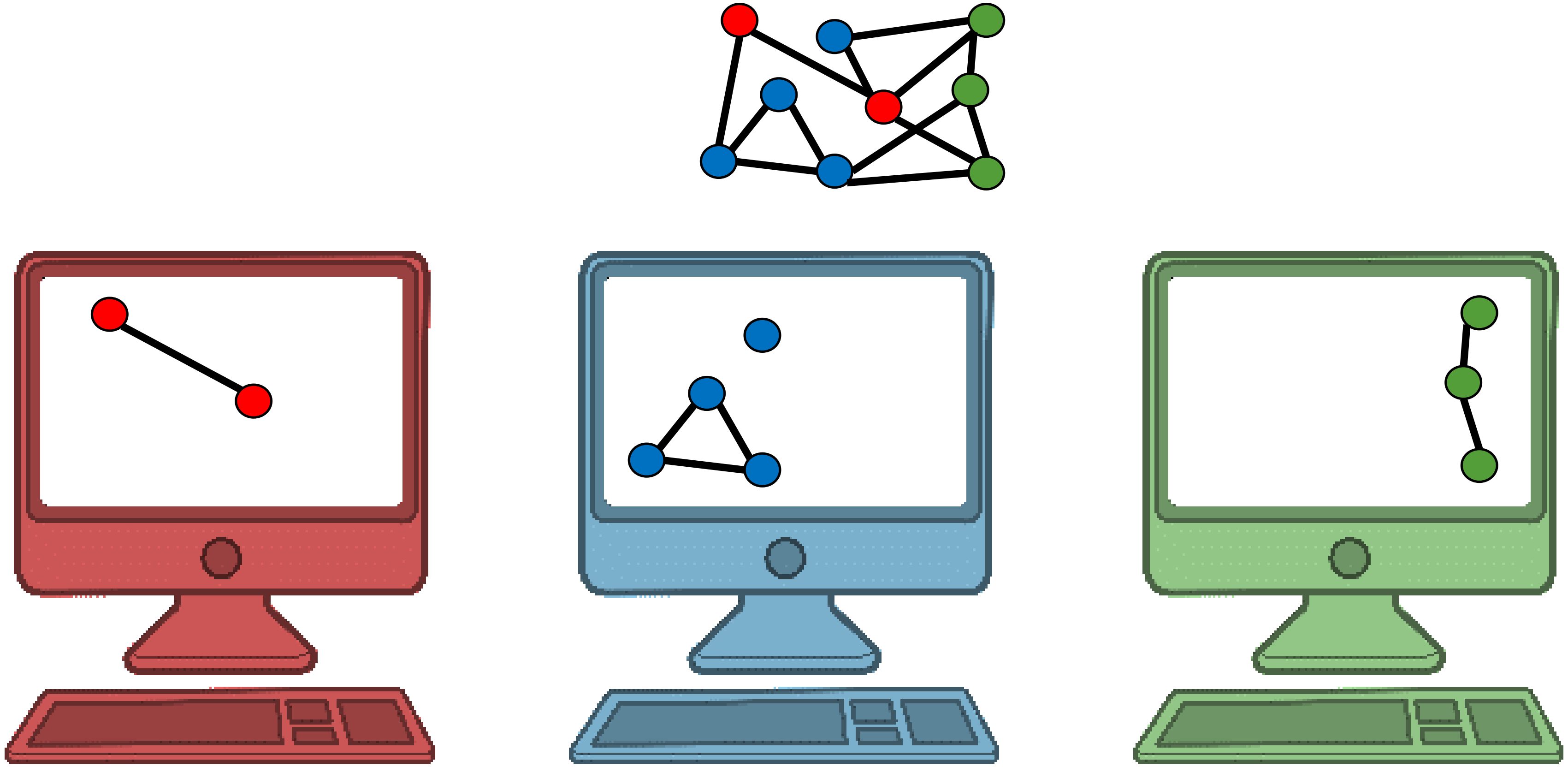
Δ = maximum degree



Random vertex partitioning

$\sqrt{\Delta}$ colors/machines

Δ = maximum degree

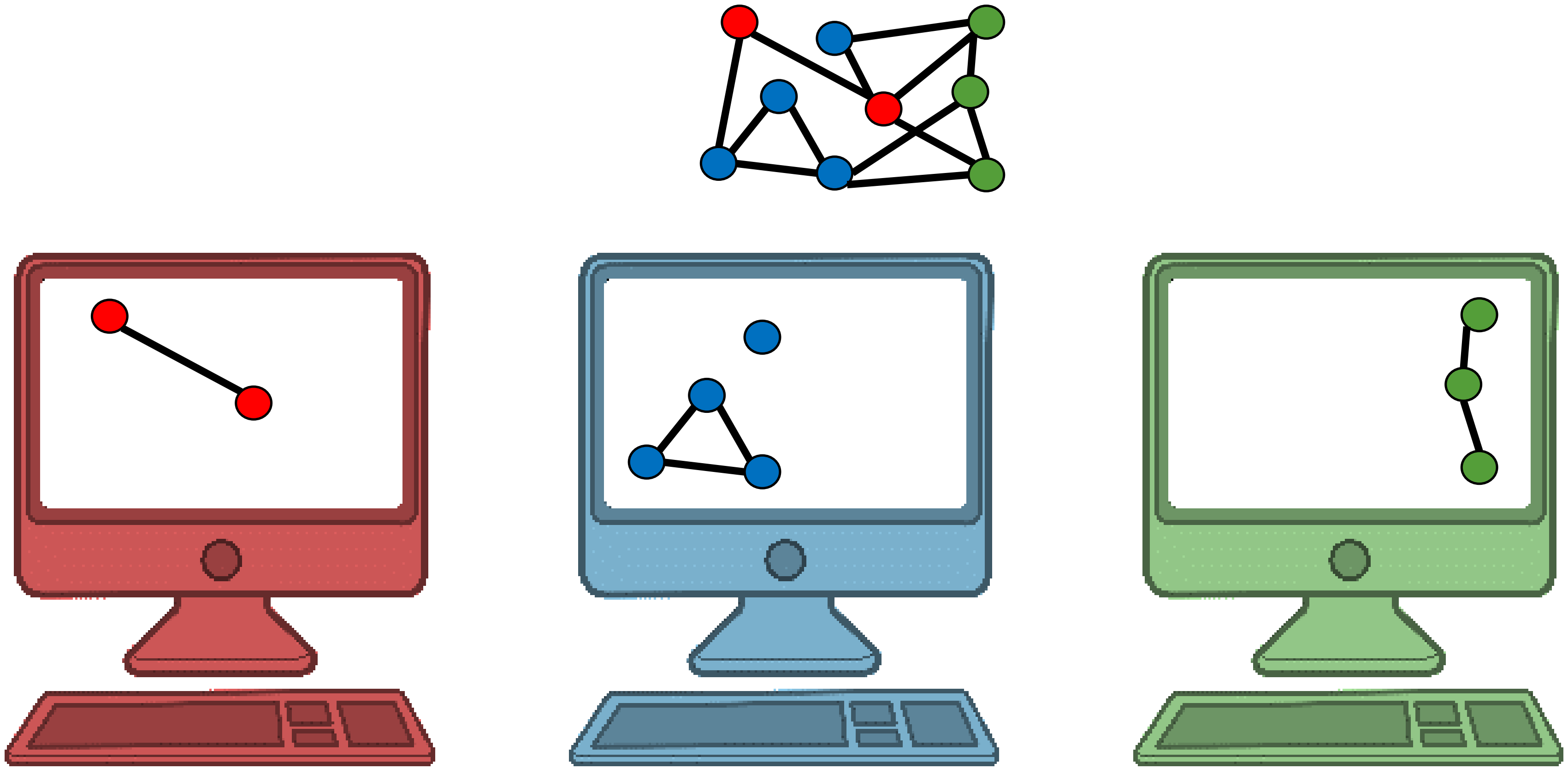


Random vertex partitioning

$\sqrt{\Delta}$ colors/machines

Δ = maximum degree

Why $\sqrt{\Delta}$ colors/machines?



E[edges on Machine i]

$$= \sum_{e \in E} \Pr[\text{e is on Machine i}]$$

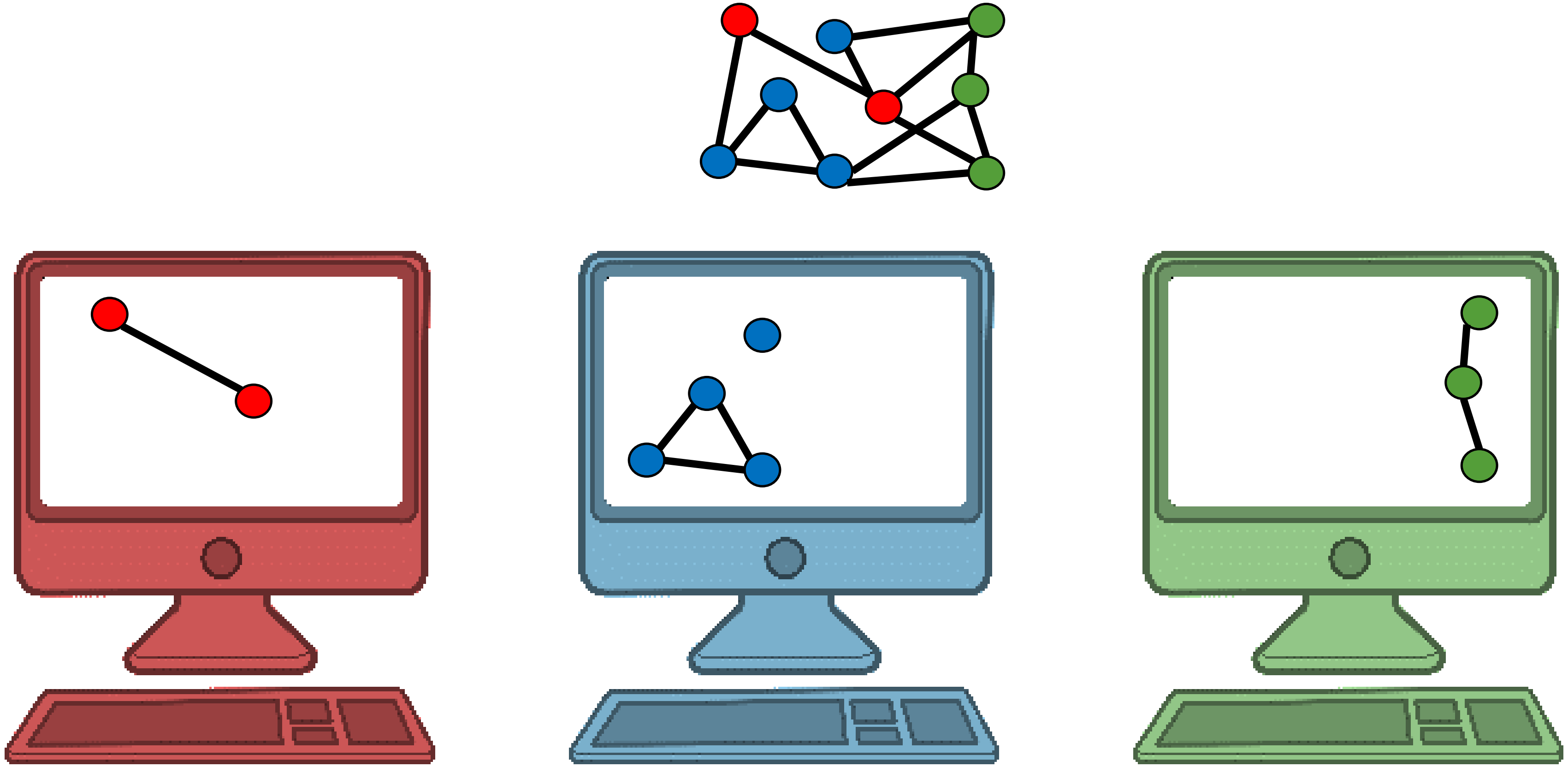
Random vertex partitioning

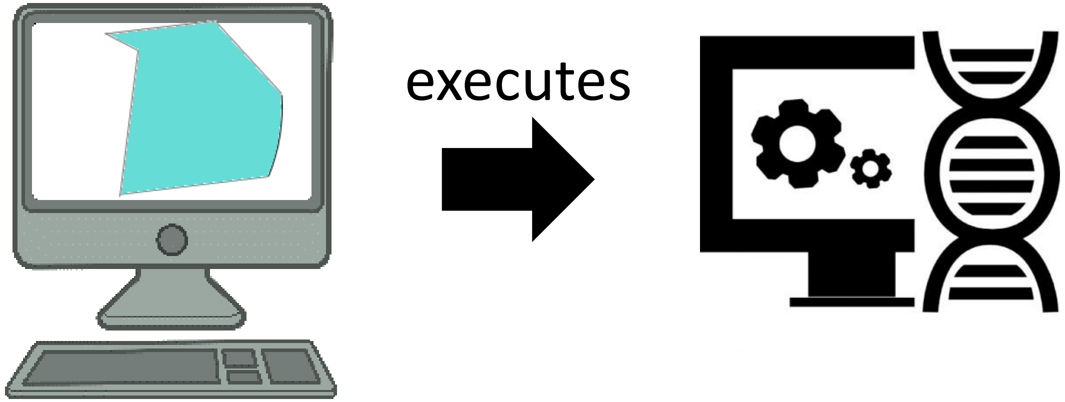
$\sqrt{\Delta}$ colors/machines

Δ = maximum degree

Why $\sqrt{\Delta}$ colors/machines?

$$\text{E[edges on Machine i]}$$
$$= \sum_{e \in E} \text{Pr[e is on Machine i]}$$
$$\leq n\Delta \frac{1}{(\sqrt{\Delta})^2} = n$$

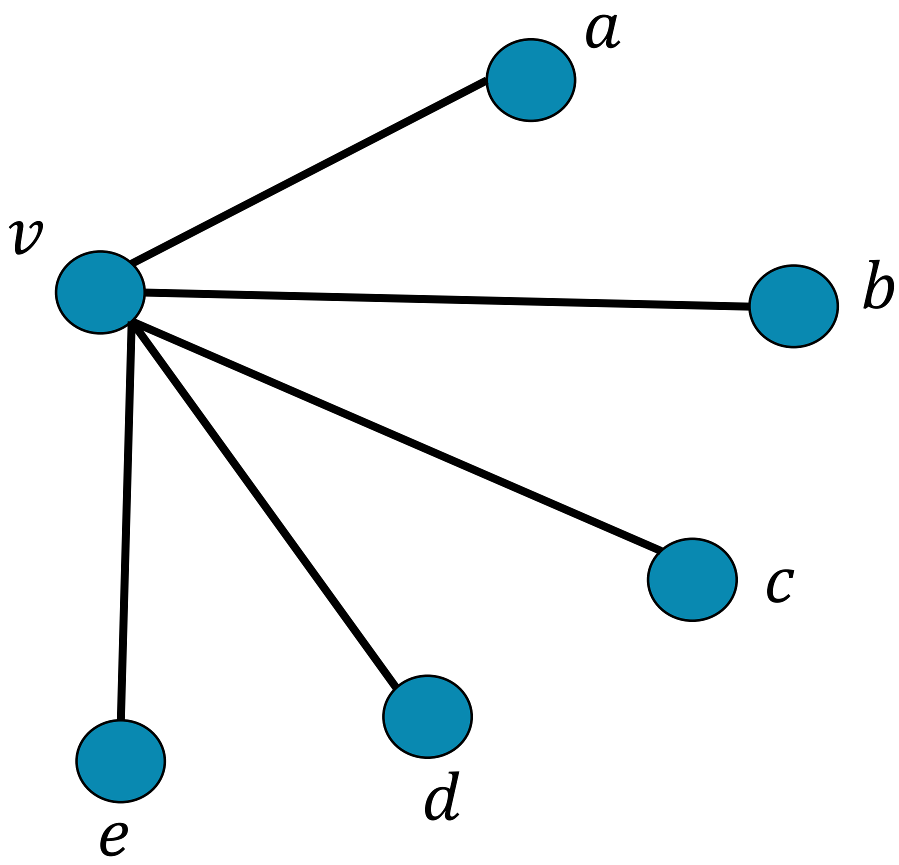




What local algorithm to use?

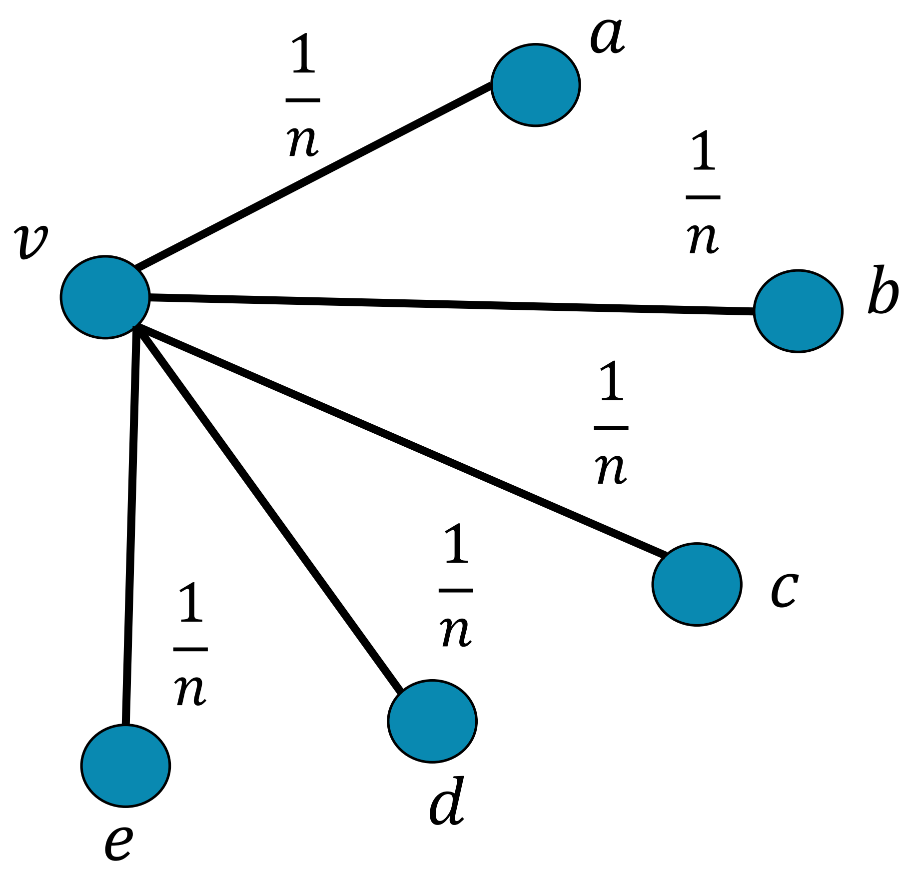
Greedy fractional matching

(CENTRALIZED)



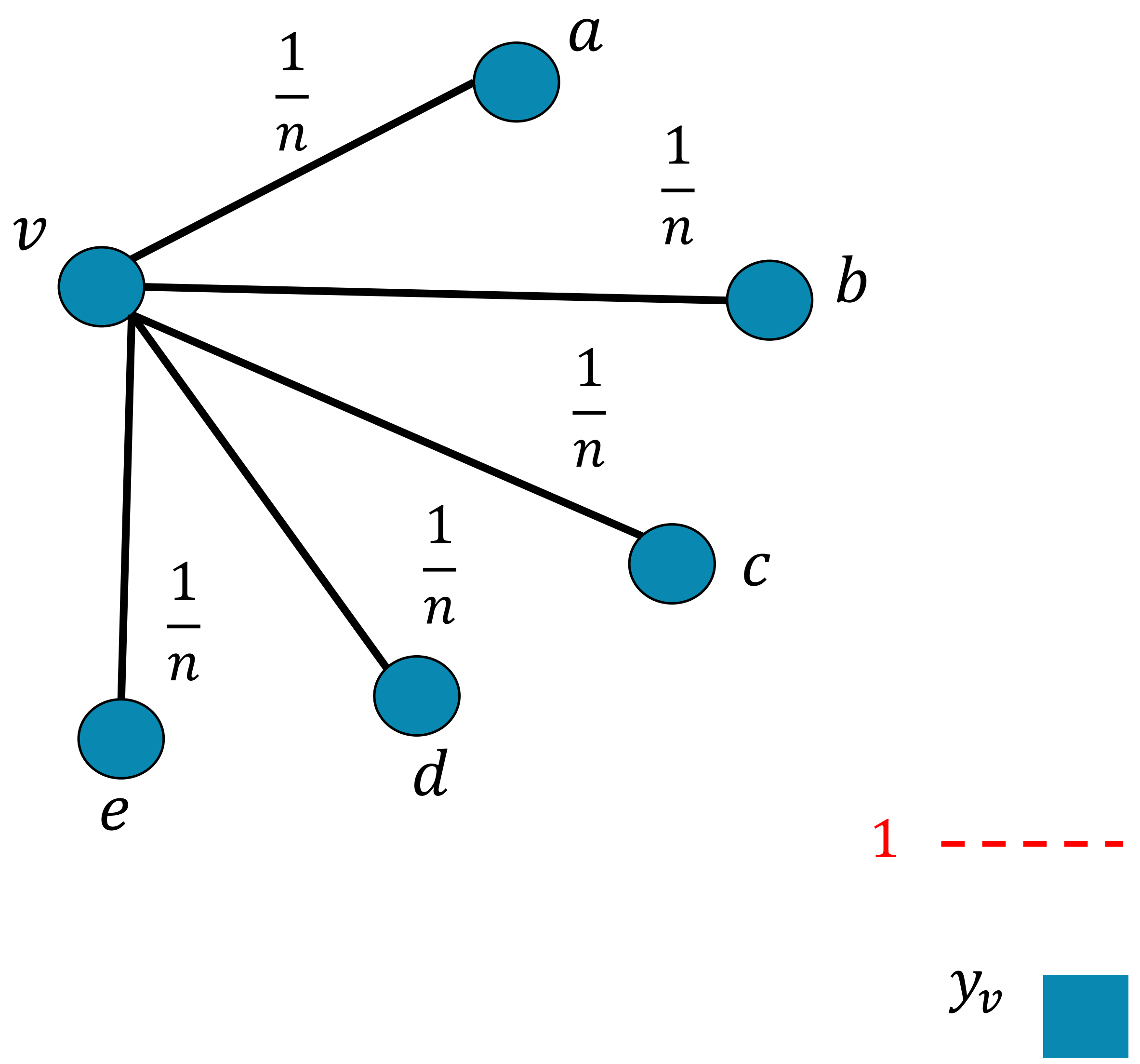
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$



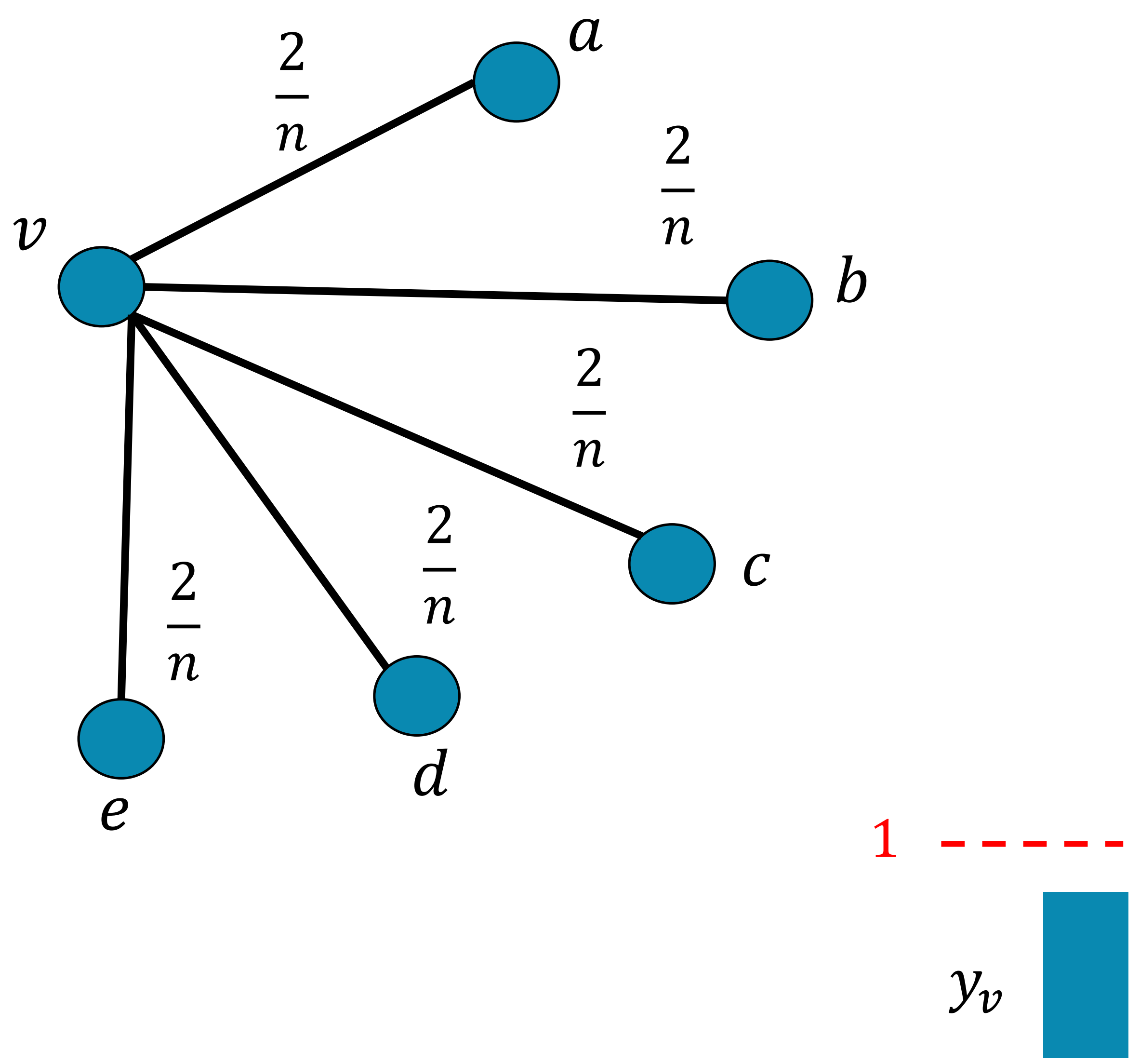
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$



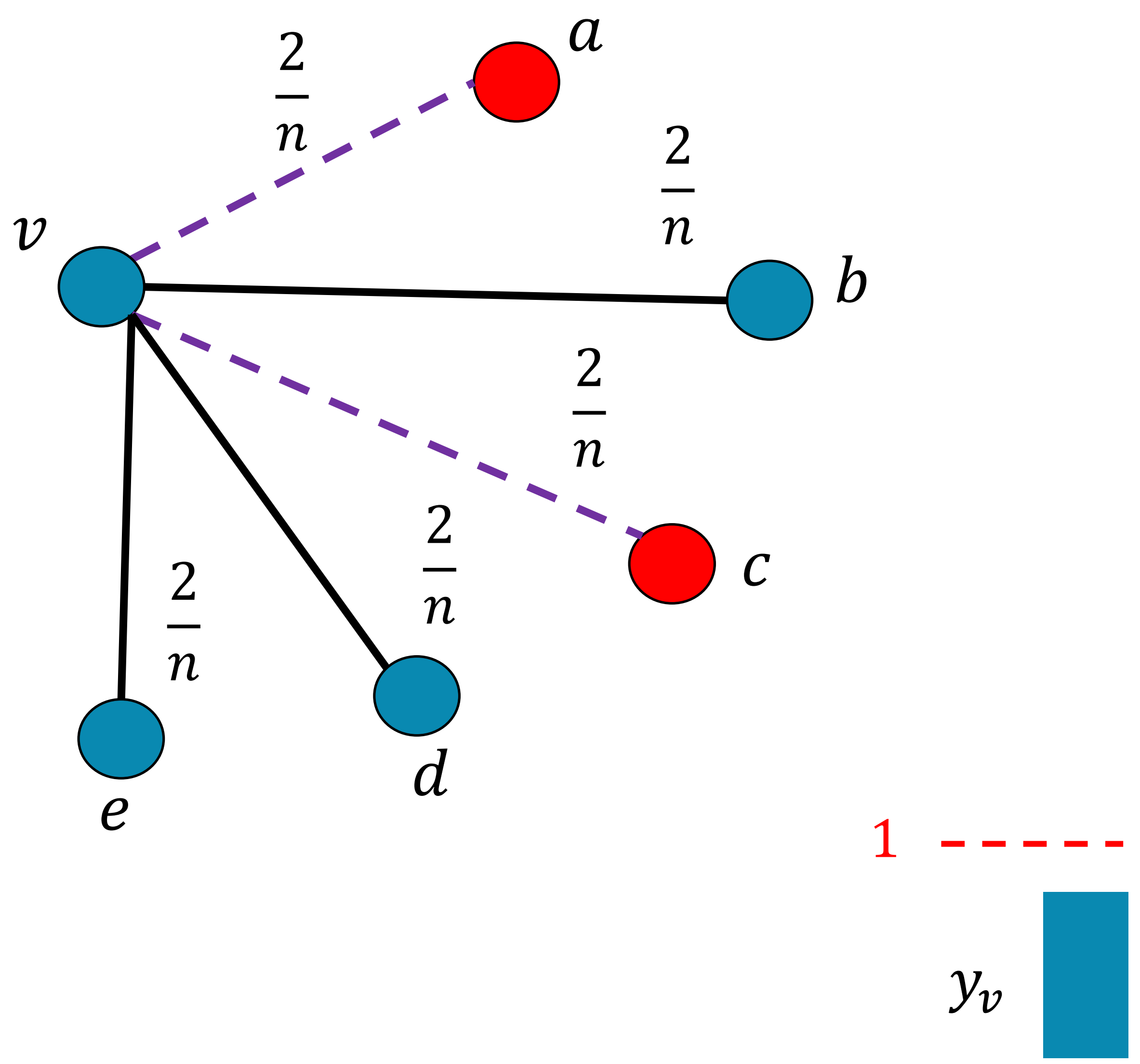
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$



Greedy fractional matching
(CENTRALIZED)

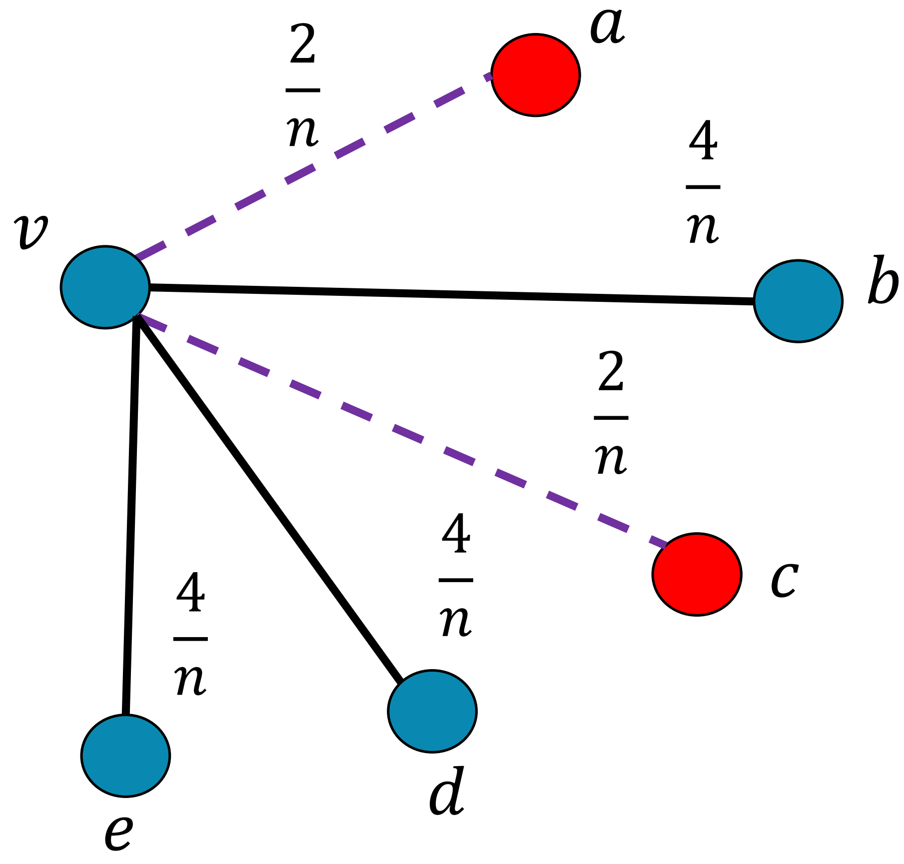
1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$

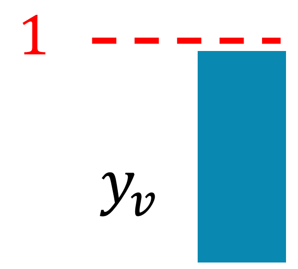


Greedy fractional matching

(CENTRALIZED)

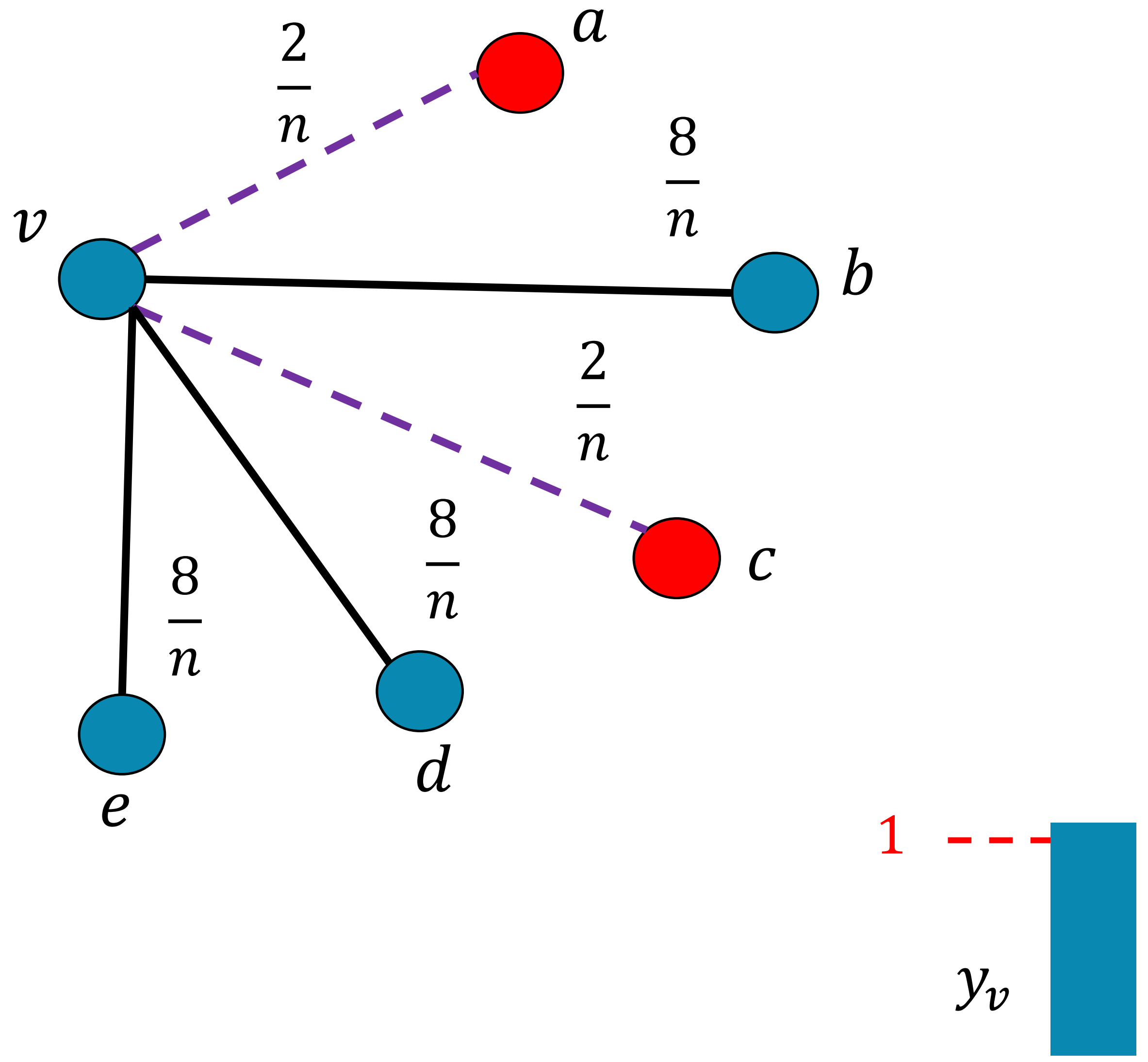
1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$





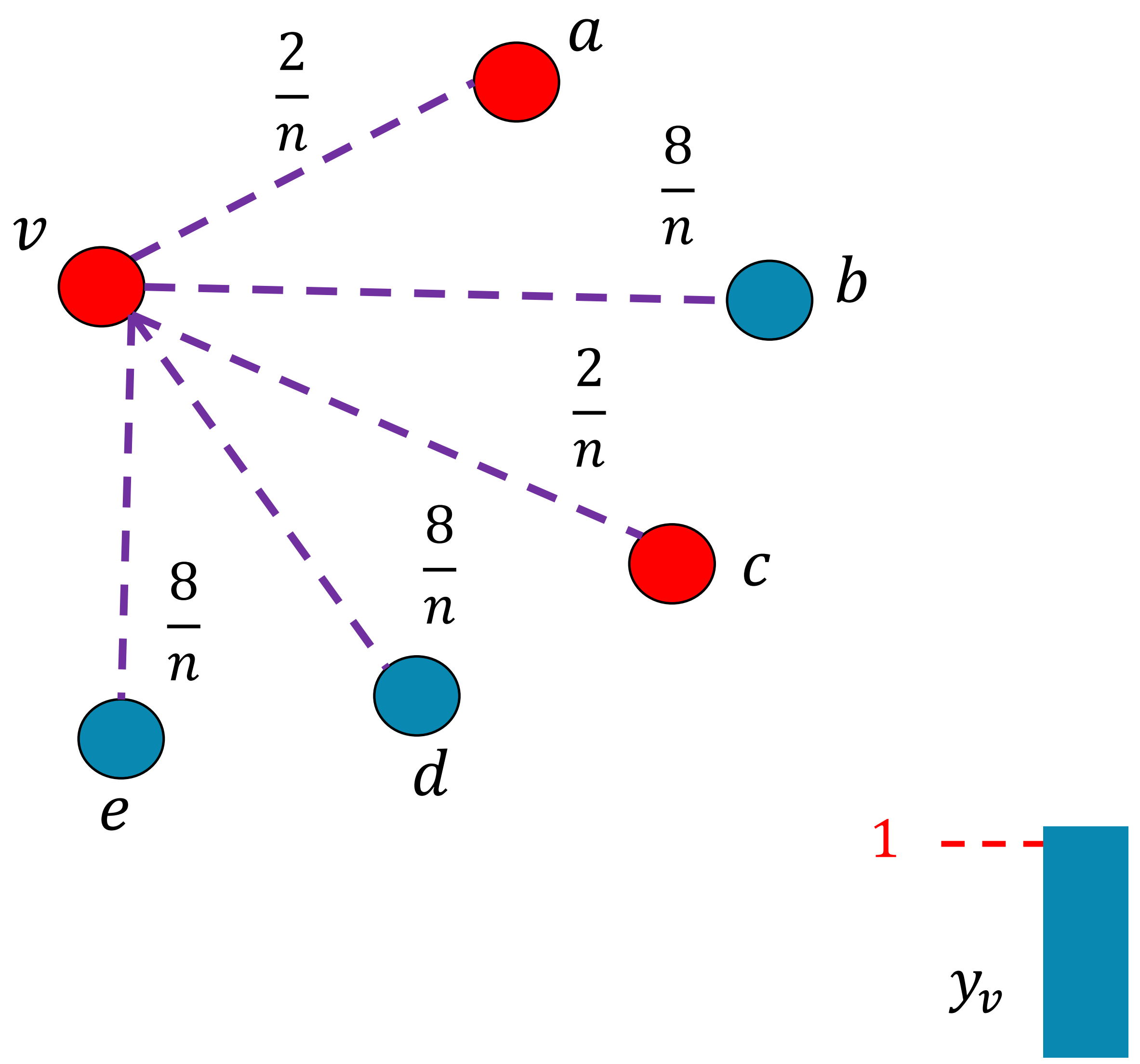
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$



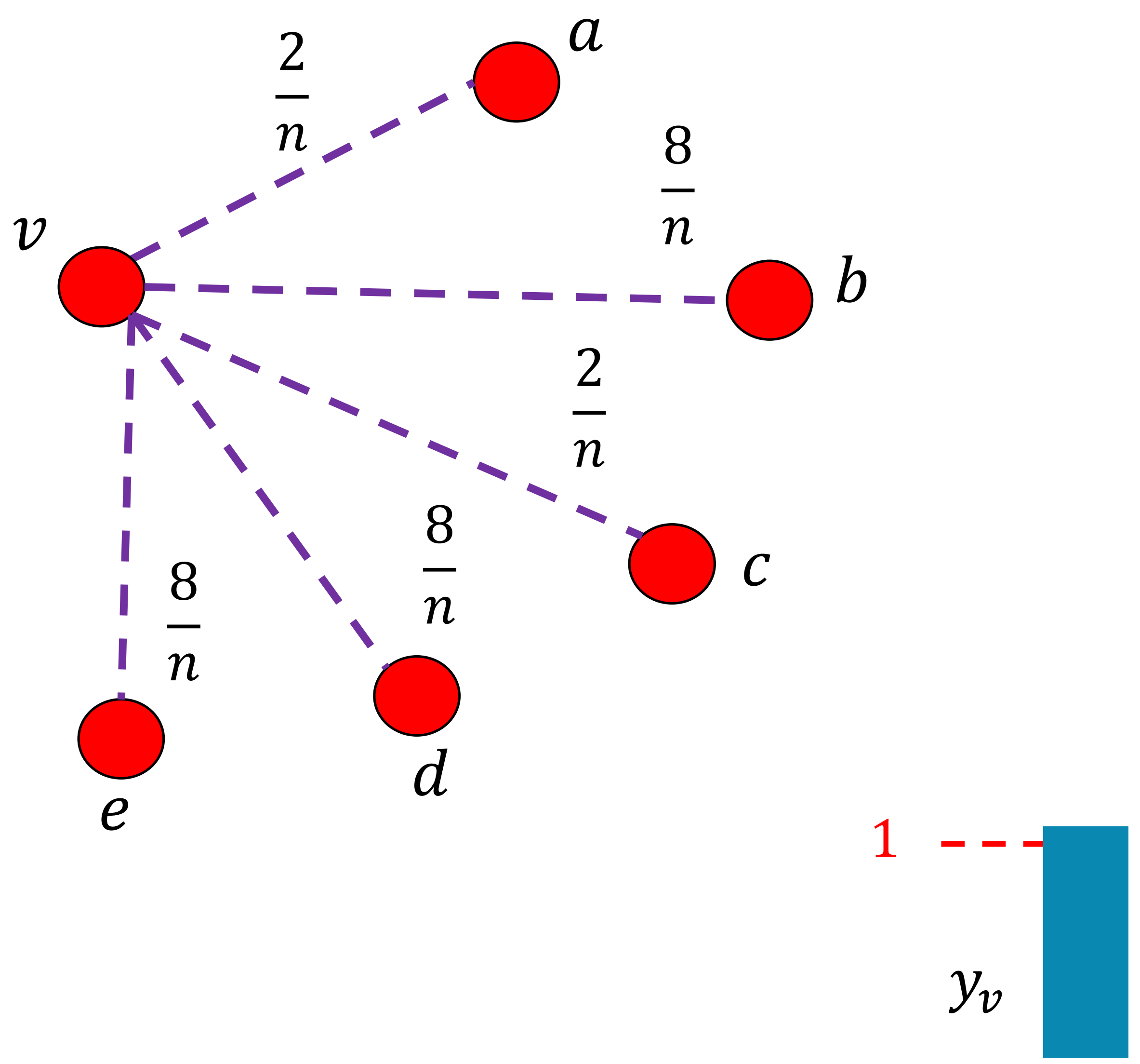
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$



Greedy fractional matching
(CENTRALIZED)

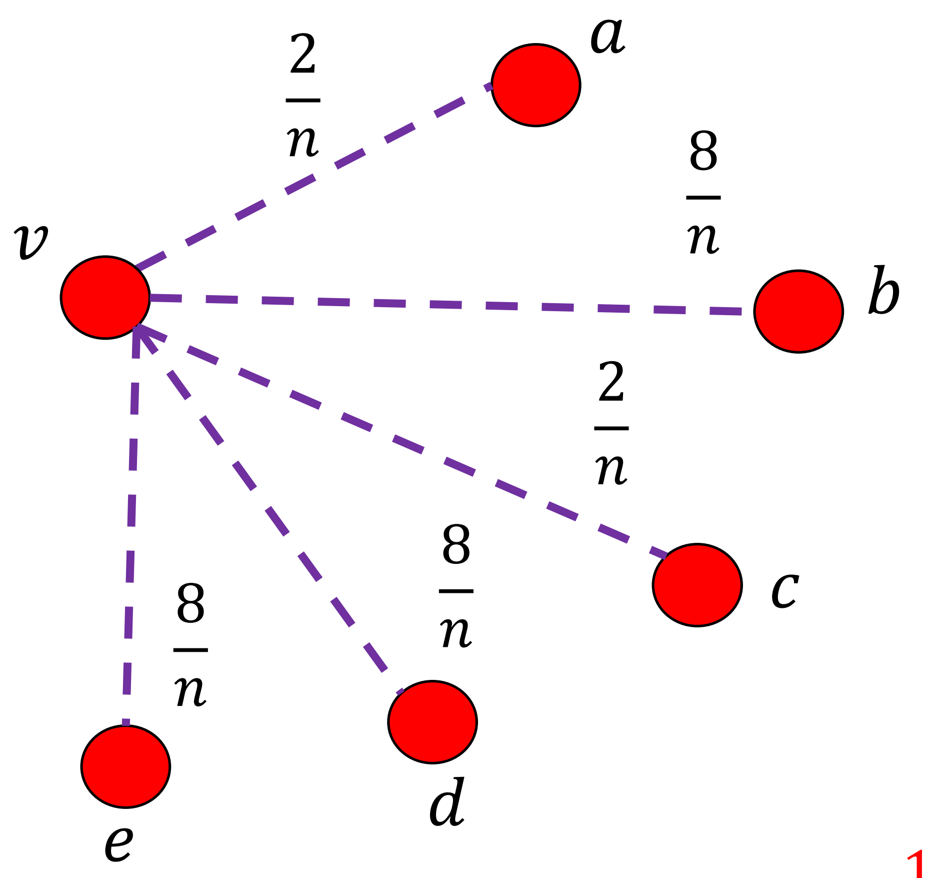
1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

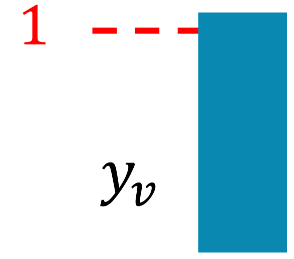


Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $\mathrm{y_v} = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

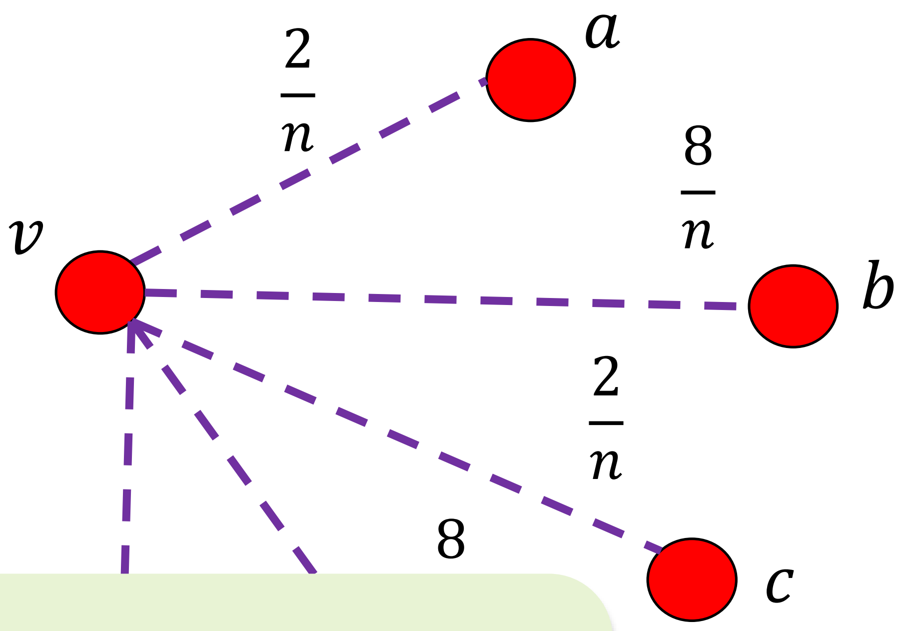
Observations:
- 4-approximate
- There are O(log n) until-loop iterations
- x_e can be deduced from when the endpoints of e cross the threshold





Greedy fractional matching
(CENTRALIZED)

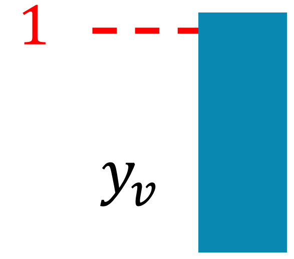
1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractiona



Can be implemented in O(log n) rounds in LOCAL and MPC.

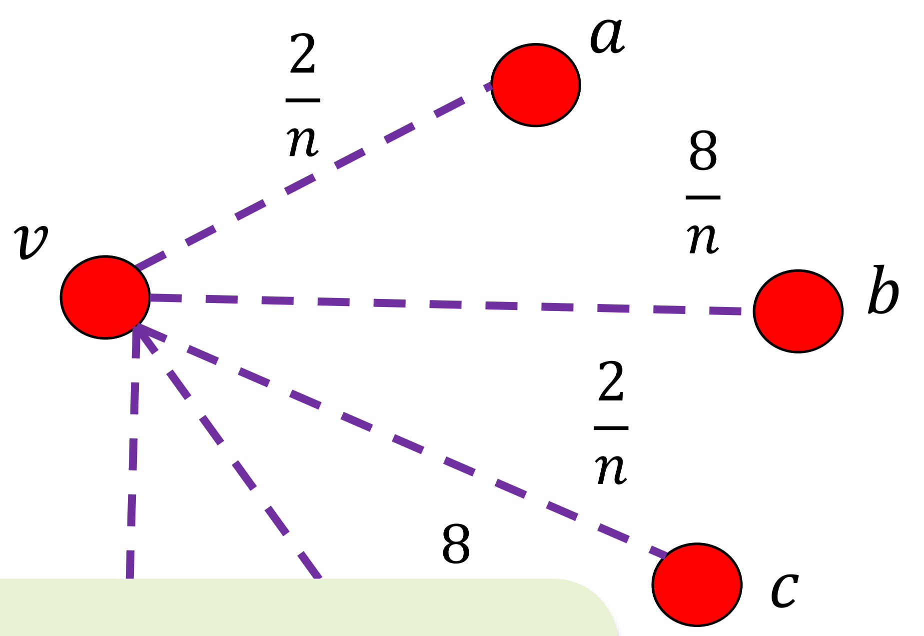
Observations:

- 4-approximate
- There are O(log n) until-loop iterations
- x_e can be deduced from when the endpoints of e cross the threshold



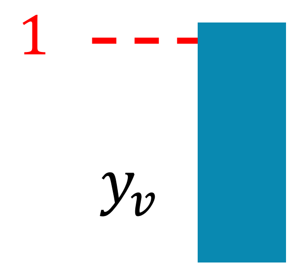
Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_\mathrm{v} = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractiona



Can be implemented in O(log n) rounds in LOCAL and MPC.
Can we implement it in **O(1) MPC rounds**?

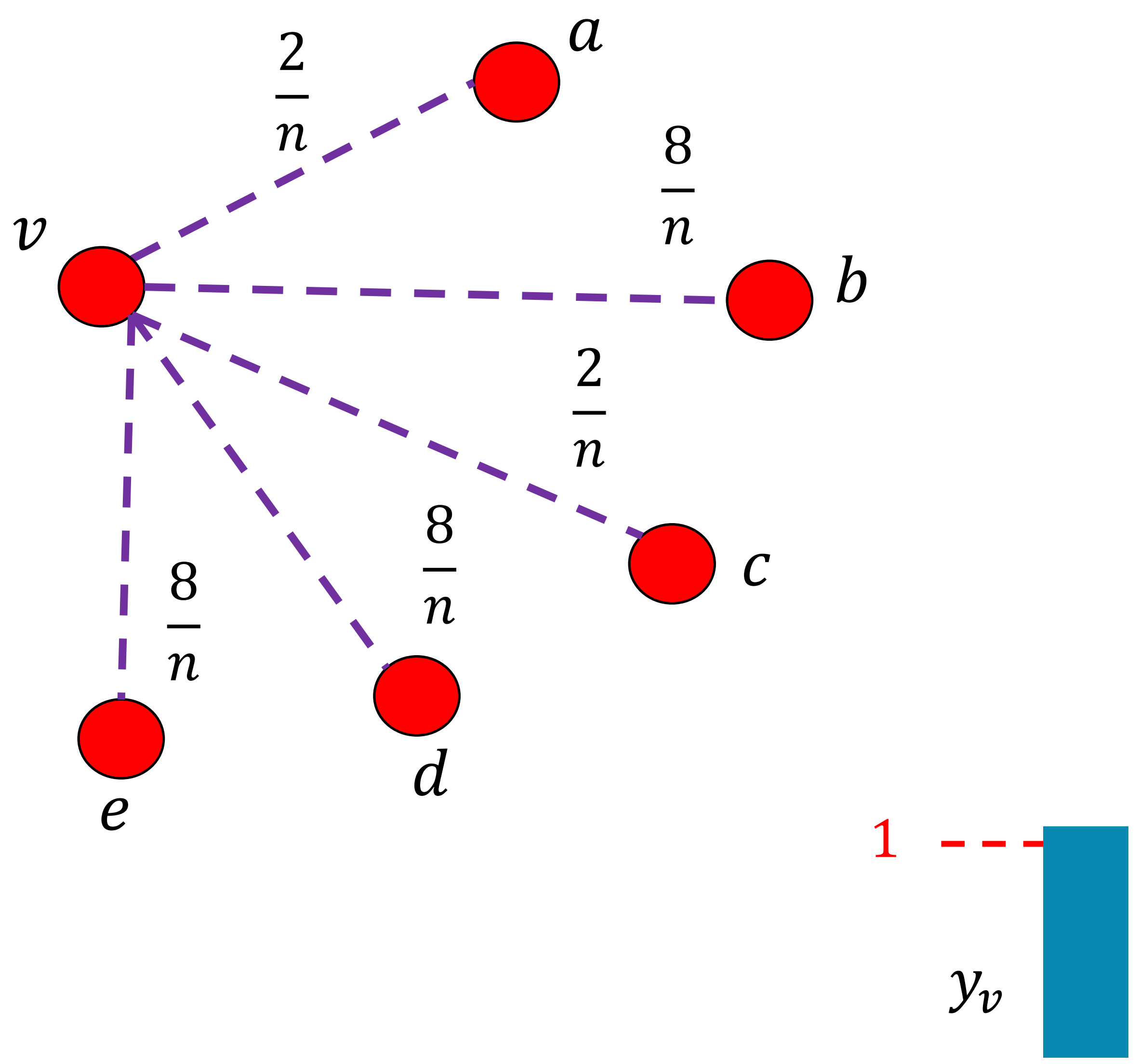
Observations:
- 4-approximate
- There are O(log n) until-loop iterations
- x_e can be deduced from when the endpoints of e cross the threshold



Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

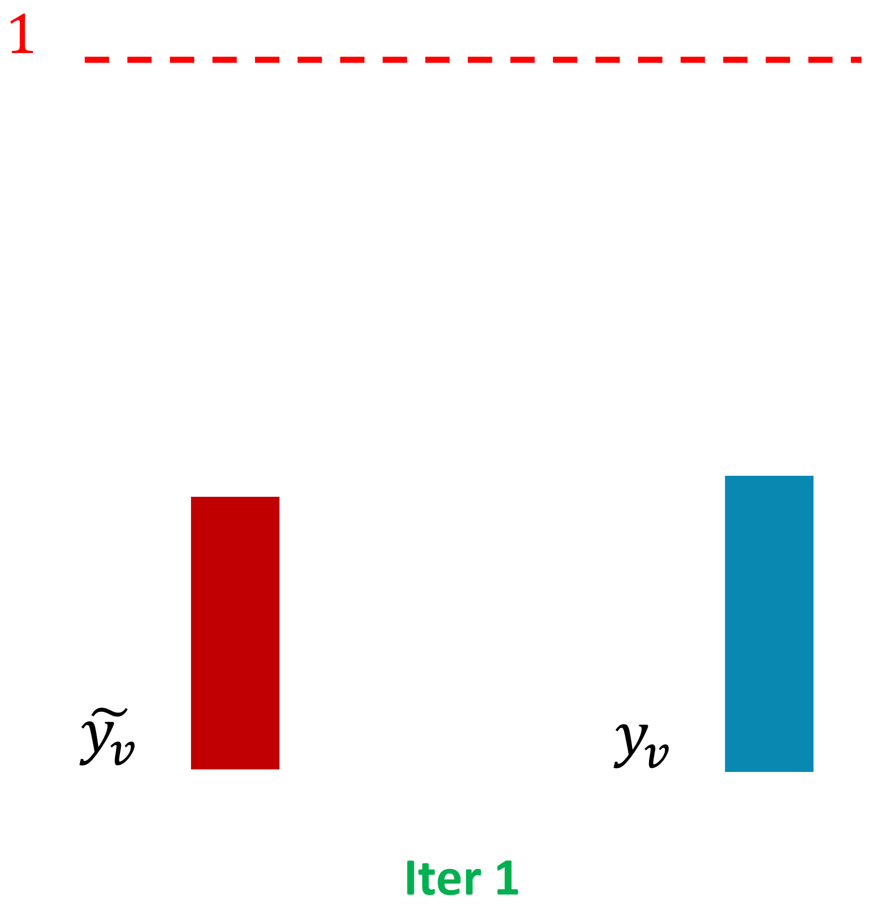
MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.

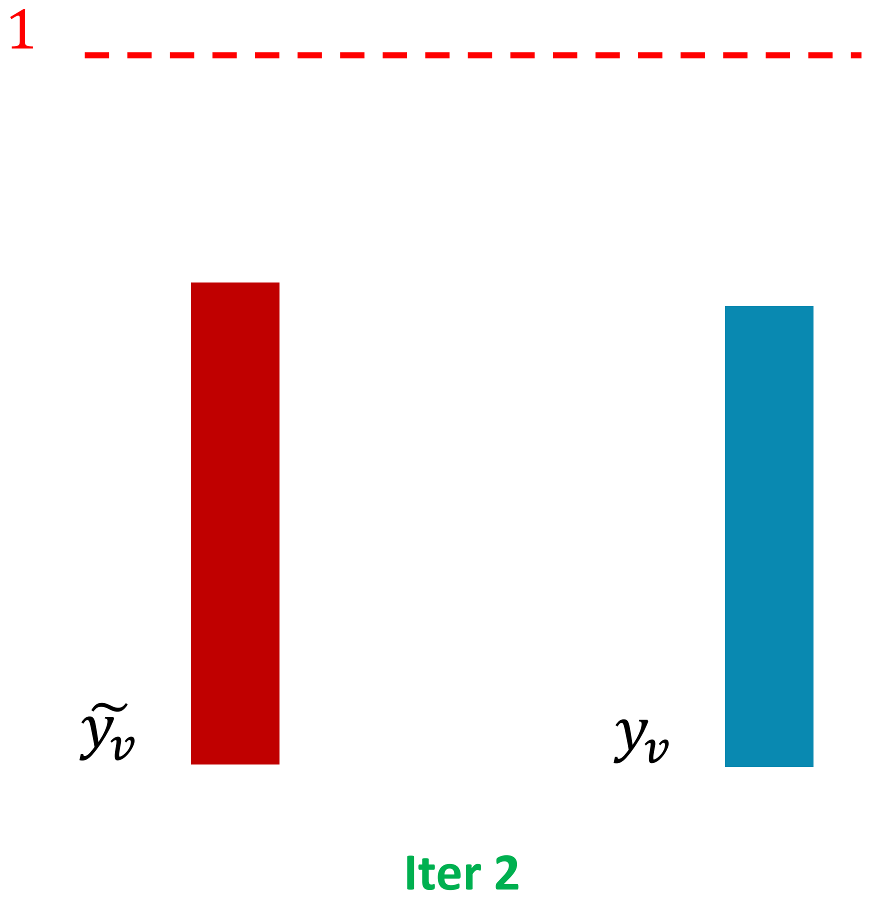


Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:

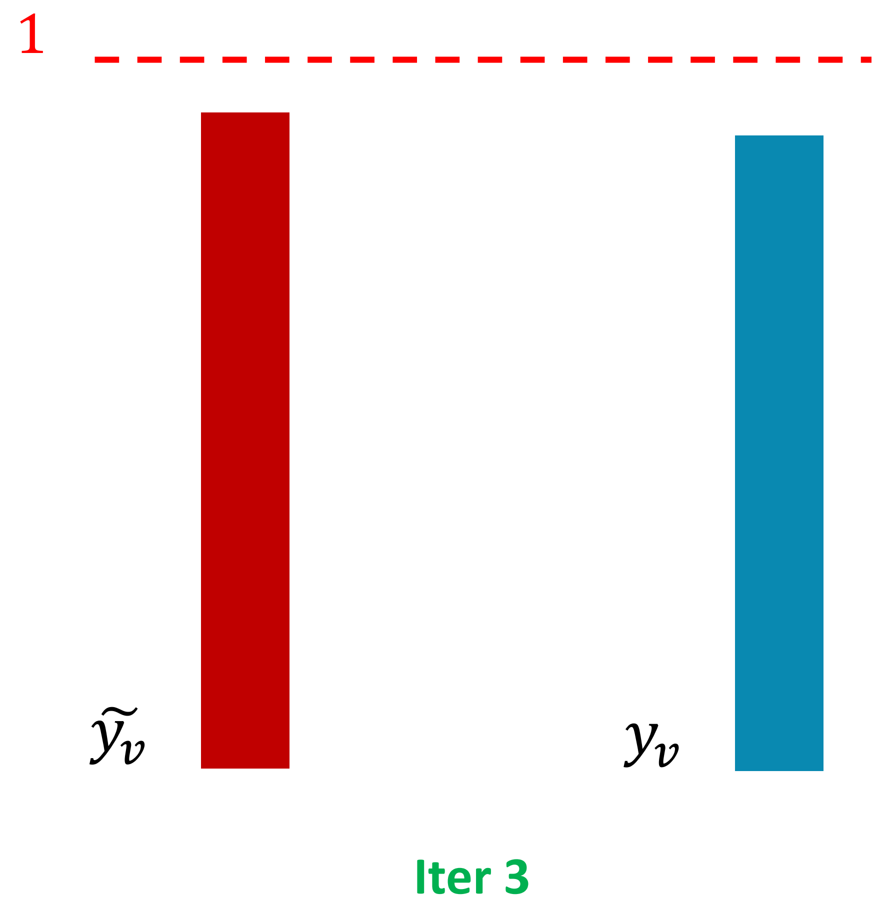
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $\mathrm{y_v} = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

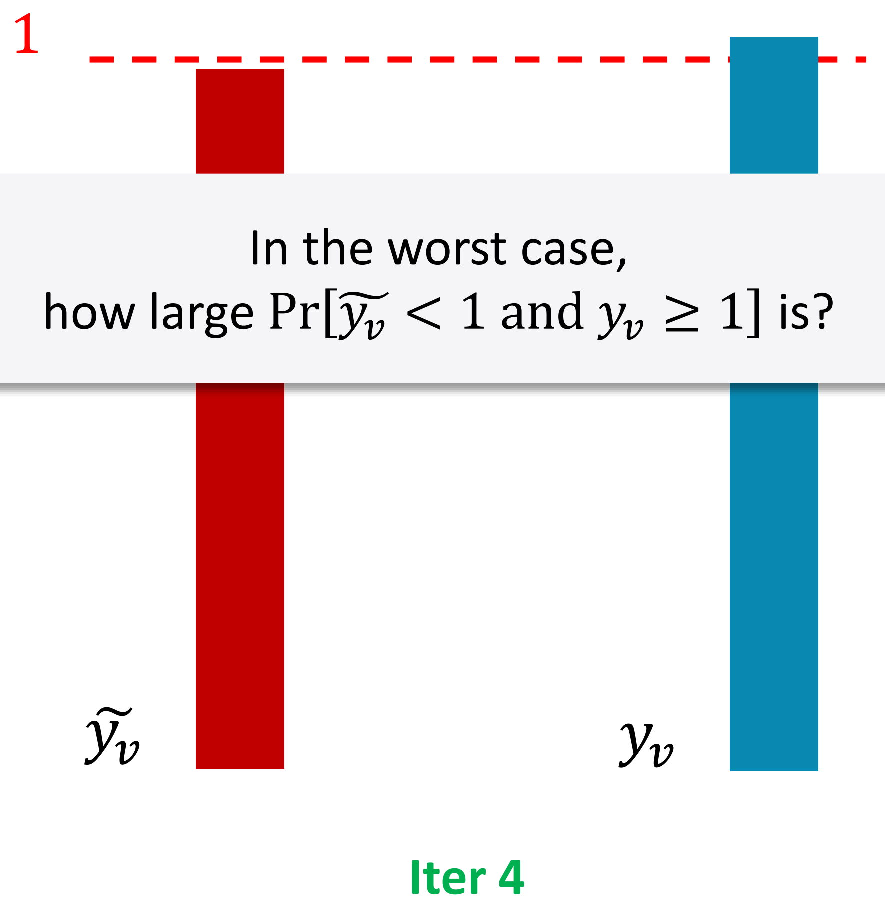
MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
  (A) **Freeze** edges incident to v for which $\mathrm{y_v} = \sum_{e \in N(v)} x_e \geq 1$
  (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.

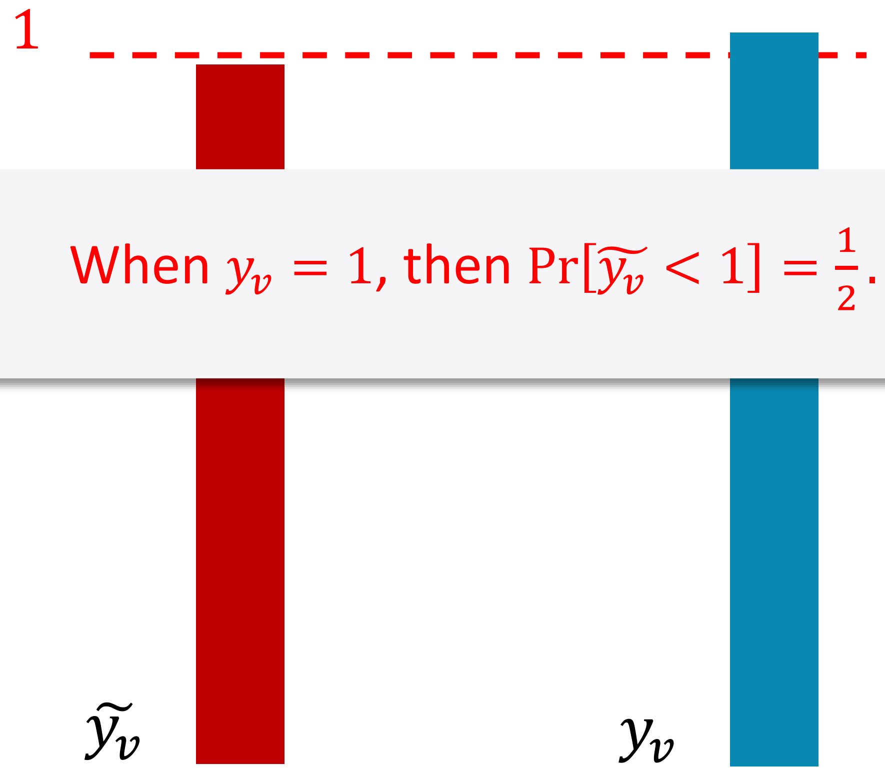


In the worst case,
how large $\Pr[\widetilde{y_v} < 1 \text{ and } y_v \geq 1]$ is?

Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



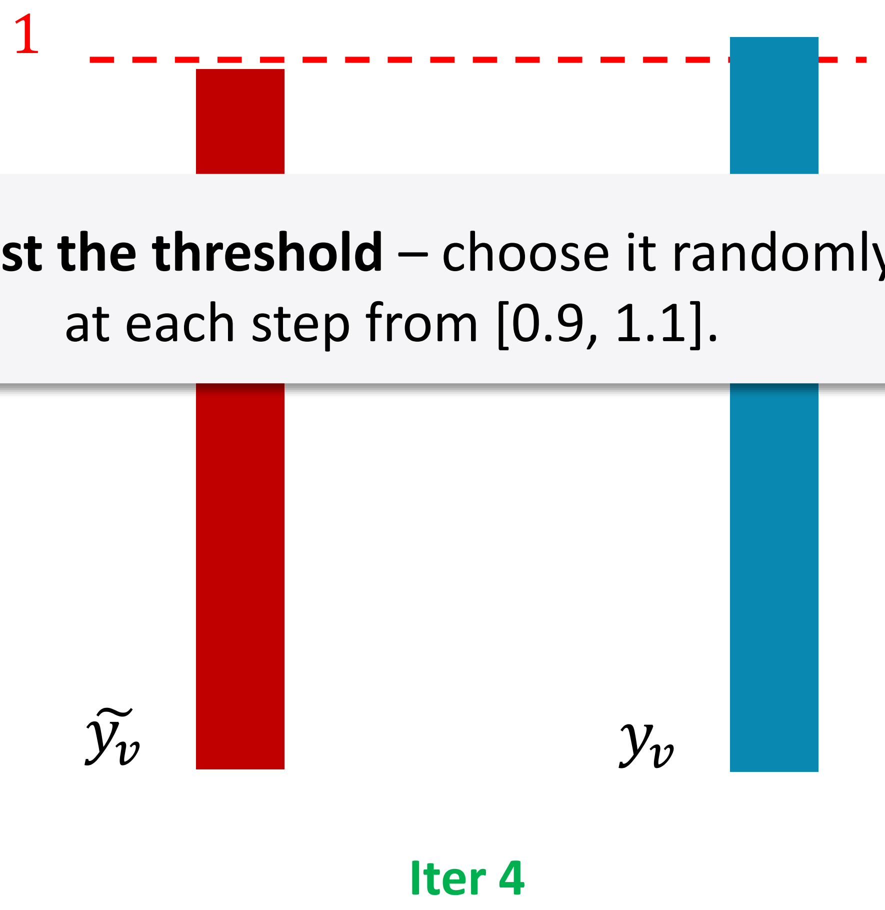
When $y_v = 1$, then $\Pr[\widetilde{y_v} < 1] = \frac{1}{2}$.

Iter 4

Greedy fractional matching
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:
- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



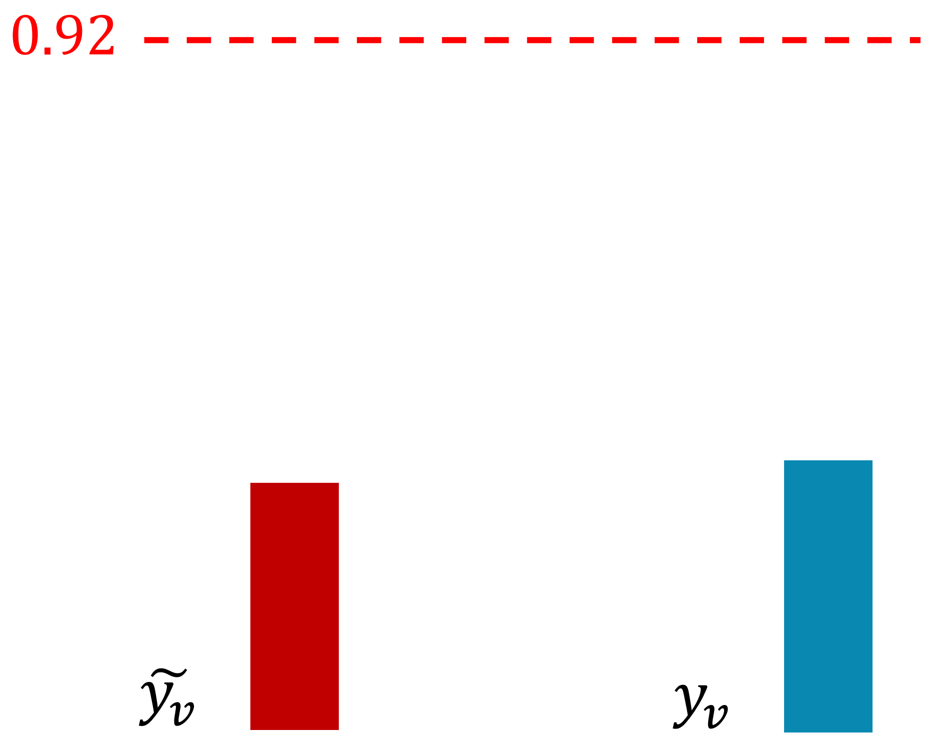
Adjust the threshold – choose it randomly at each step from [0.9, 1.1].

Greedy fractional matching with random thresholding (CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq Rnd(0.9, 1.1)$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:

- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.

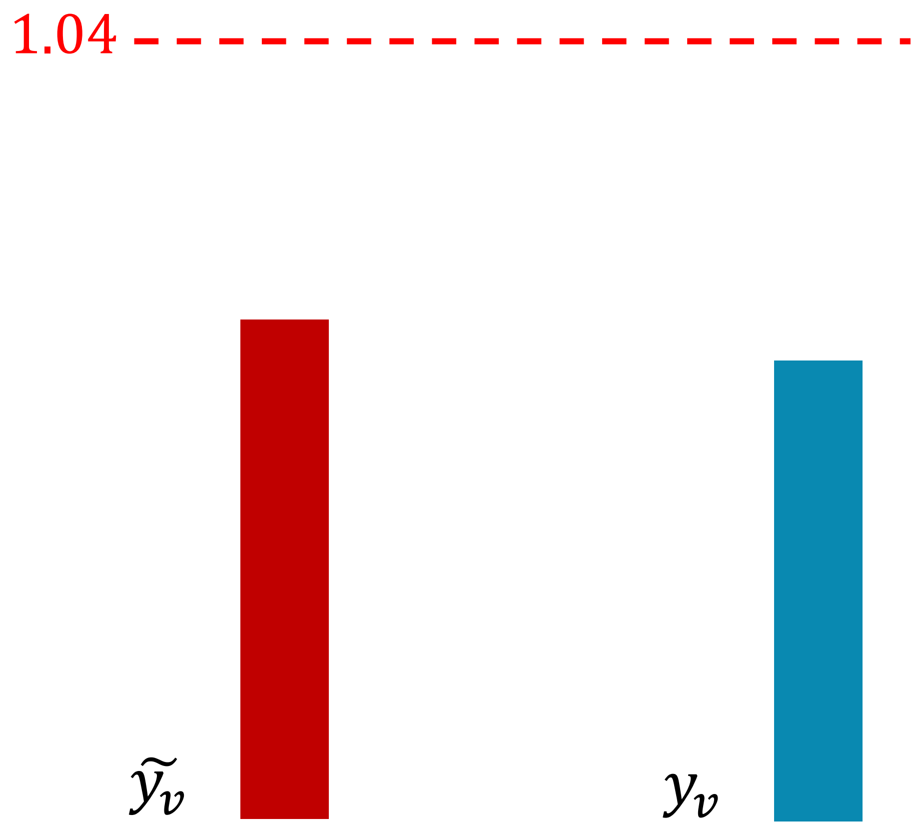


Greedy fractional matching with random thresholding (CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq Rnd(0.9, 1.1)$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:

- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.

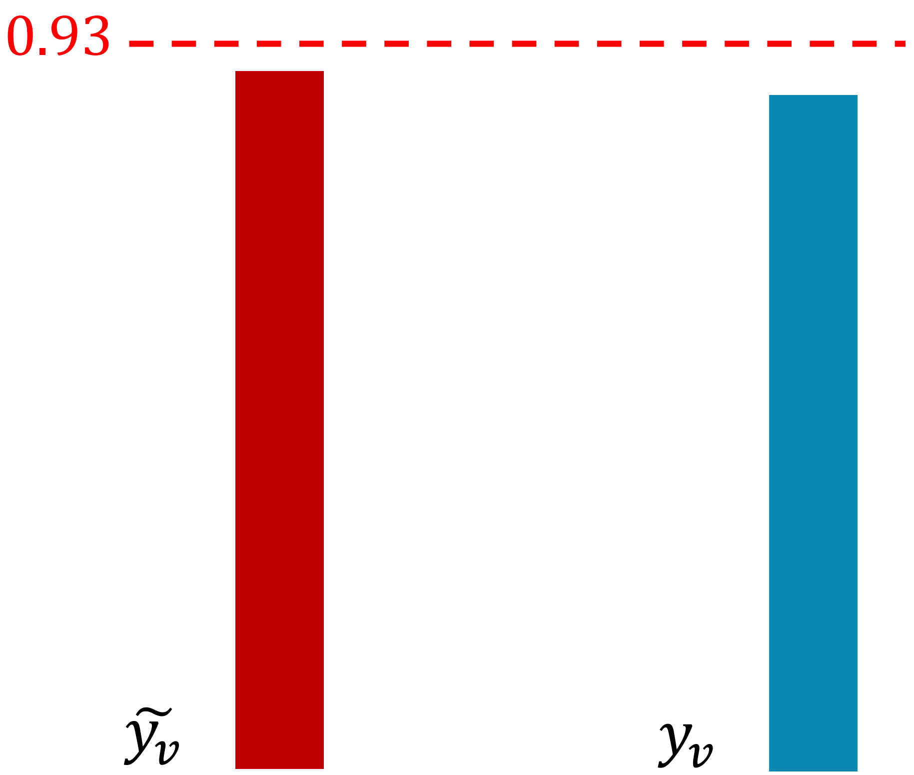


Greedy fractional matching with random thresholding (CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq Rnd(0.9, 1.1)$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:

- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.

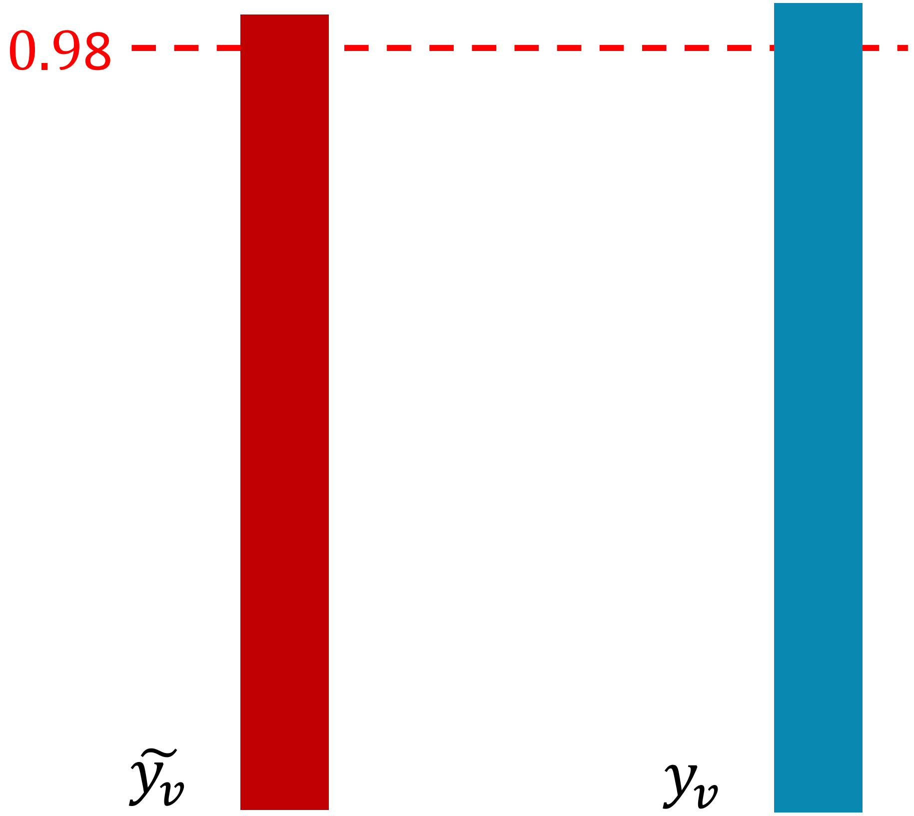


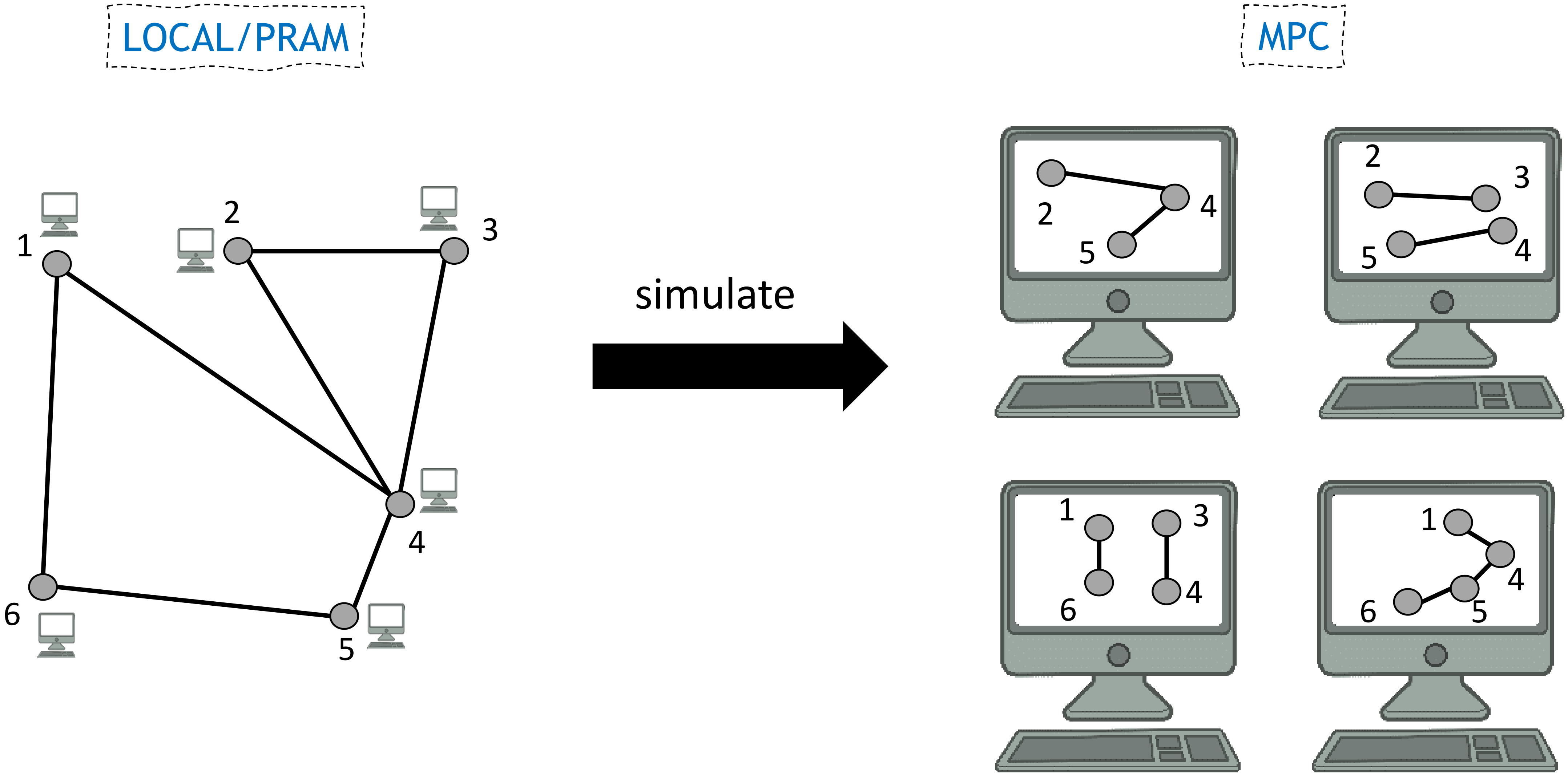
Greedy fractional matching with random thresholding
(CENTRALIZED)

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq Rnd(0.9, 1.1)$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching

MPC Simulation Idea:

- Sample a subgraph and *estimate* y_v.
- Use the estimates to **freeze** the edges.



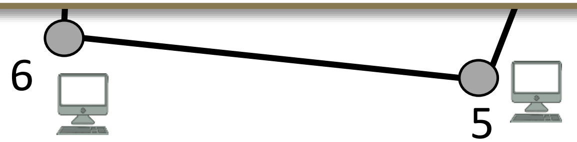


Algorithm: A
Rounds: T

Algorithm: ≈A
Rounds: o(T)

LOCAL/PRAM

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which $y_v = \sum_{e \in N(v)} x_e \geq 1$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching



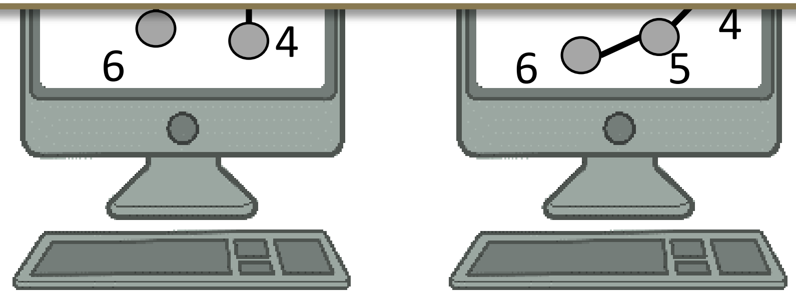
Algorithm: A

Rounds: T

But what is o(T)?

MPC

1. Initially, for every $e \in E$, set $x_e = \frac{1}{n}$
2. Until each edge is **frozen**:
   (A) **Freeze** edges incident to v for which an *estimate* of $y_v = \sum_{e \in N(v)} x_e \geq Rnd(0.9, 1.1)$
   (B) For each unfrozen edge, set $x_e = 2 \cdot x_e$
3. Output $\frac{x}{2}$ as a fractional matching



Algorithm: ≈A

Rounds: o(T)

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\widetilde{y_v} = \sqrt{n} \sum_{e \in N_{\boldsymbol{locally}}(v)} x_e = \frac{1}{\sqrt{n}} \left| N_{\boldsymbol{locally}}(v) \right|$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\widetilde{y_v} = \sqrt{n} \sum_{e \in N_{\boldsymbol{locally}}(v)} x_e = \frac{1}{\sqrt{n}} \left|N_{\boldsymbol{locally}}(v)\right|$
- $E\left[\left|N_{\boldsymbol{locally}}(v)\right|\right] = \frac{\mathrm{d}_v}{\sqrt{n}} \geq n^{0.4}$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\widetilde{y_v} = \sqrt{n} \sum_{e \in N_{\boldsymbol{locally}}(v)} x_e = \frac{1}{\sqrt{n}} |N_{\boldsymbol{locally}}(v)|$
- $E\left[|N_{\boldsymbol{locally}}(v)|\right] = \frac{\mathrm{d}_v}{\sqrt{n}} \geq n^{0.4}$
- With high prob: $\left| |N_{\boldsymbol{locally}}(v)| - \frac{\mathrm{d_v}}{\sqrt{n}} \right| \leq n^{0.3}$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\tilde{y}_v$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\tilde{y}_v = \sqrt{n} \sum_{e \in N_{\boldsymbol{locally}}(v)} x_e = \frac{1}{\sqrt{n}} |N_{\boldsymbol{locally}}(v)|$
- $E\left[|N_{\boldsymbol{locally}}(v)|\right] = \frac{\mathrm{d}_v}{\sqrt{n}} \geq n^{0.4}$
- With high prob: $\left| |N_{\boldsymbol{locally}}(v)| - \frac{\mathrm{d_v}}{\sqrt{n}} \right| \leq n^{0.3}$
- With high prob: $|y_v - \tilde{y}_v| \leq n^{-0.2}$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\widetilde{y_v} = \sqrt{n} \sum_{e \in N_{\boldsymbol{locally}}(v)} x_e = \frac{1}{\sqrt{n}} |N_{\boldsymbol{locally}}(v)|$
- $E\left[|N_{\boldsymbol{locally}}(v)|\right] = \frac{\mathrm{d}_v}{\sqrt{n}} \geq n^{0.4}$
- With high prob: $\left| |N_{\boldsymbol{locally}}(v)| - \frac{\mathrm{d_v}}{\sqrt{n}} \right| \leq n^{0.3}$
- With high prob: $|y_v - \widetilde{y_v}| \leq n^{-0.2}$

Before: When $y_v = 1$, $\Pr[\widetilde{y_v} < 1] = \frac{1}{2}$.

What is the probability that a random threshold "cuts" between $\widetilde{y_v}$ and y_v?

How much random thresholding gains? I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

Setup:
- $\sqrt{n}$ colors/machines
- Random vertex partitioning
- ***Goal***: $\widetilde{y_v}$ and y_v cross the threshold at the same time!

Consider a vertex v with $\mathrm{d}_v \geq n^{0.9}$, and Iter 1

- $y_v = \frac{\mathrm{d}_v}{n}$
- $\widetilde{y_v} = \sqrt{n} \sum_{e \in N_{\textbf{locally}}(v)} x_e = \frac{1}{\sqrt{n}} |N_{\textbf{locally}}(v)|$
- $E\left[|N_{\textbf{locally}}(v)|\right] = \frac{\mathrm{d}_v}{\sqrt{n}} \geq n^{0.4}$
- With high prob: $\left||N_{\textbf{locally}}(v)| - \frac{\mathrm{d_v}}{\sqrt{n}}\right| \leq n^{0.3}$
- With high prob: $|y_v - \widetilde{y_v}| \leq n^{-0.2}$

Before: When $y_v = 1$, $\Pr[\widetilde{y_v} < 1] = \frac{1}{2}$.

What is the probability that a random threshold "cuts" between $\widetilde{y_v}$ and y_v?

$$\leq \frac{n^{-0.2}}{1.1 - 0.9}$$

How much random thresholding gains? I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

$\mathrm{d}_v \geq n^{0.9}$

Iter 1:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_1 = \frac{n^{-0.2}}{0.2}$$

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

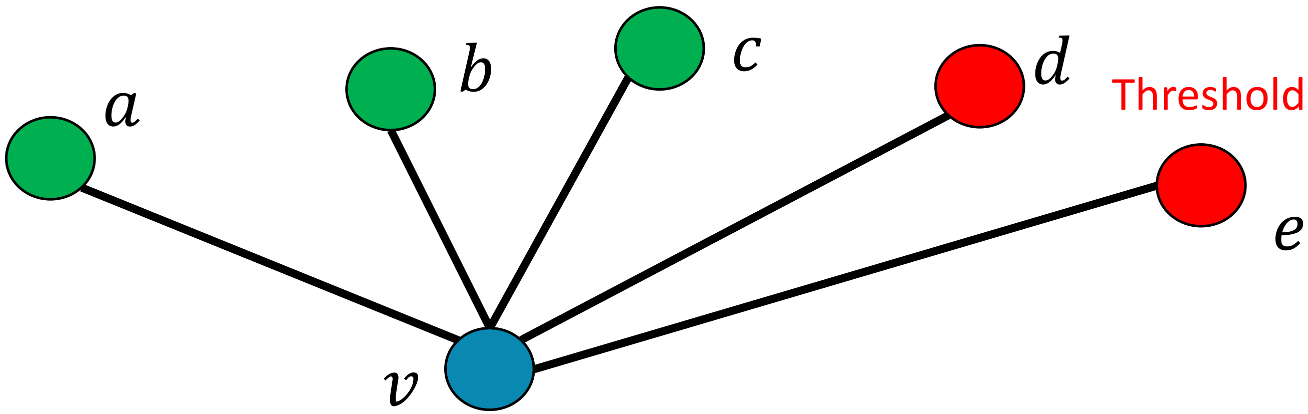
$d_v \geq n^{0.9}$

Iter 1:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_1 = \frac{n^{-0.2}}{0.2}$$

Iter 2:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_2 = \frac{O(\sigma_1) + n^{-0.2}}{0.2} \leq 10\sigma_1$$



Threshold cuts between y and $\tilde{y}$ in **Iter 1**.

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

$\mathrm{d}_v \geq n^{0.9}$

Iter 1:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_1 = \frac{n^{-0.2}}{0.2}$$

Iter 2:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_2 = \frac{O(\sigma_1) + n^{-0.2}}{0.2} \leq 10\sigma_1$$

...

Iter i:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq 10^i\sigma_1$$

We aim for $10^i\sigma_1 \leq 0.0001$.

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

$d_v \geq n^{0.9}$

Iter 1:

Pr[random threshold "cuts" between y_v and $\widetilde{y_v}$] $\leq \sigma_1 = \frac{n^{-0.2}}{0.2}$

Iter 2:

Pr[random threshold "cuts" between y_v and $\widetilde{y_v}$] $\leq \sigma_2 = \frac{O(\sigma_1) + n^{-0.2}}{0.2} \leq 10\sigma_1$

…

Iter i:

Pr[random threshold "cuts" between y_v and $\widetilde{y_v}$] $\leq 10^i \sigma_1$

We aim for $10^i \sigma_1 \leq 0.0001$.

After a constant fraction of iterations,
the probability error becomes too high.

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

$d_v \geq n^{0.9}$

Iter 1:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_1 = \frac{n^{-0.2}}{0.2}$$

Iter 2:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq \sigma_2 = \frac{O(\sigma_1) + n^{-0.2}}{0.2} \leq 10\sigma_1$$

…

Iter i:

$$\Pr[\text{random threshold "cuts" between } y_v \text{ and } \widetilde{y_v}] \leq 10^i \sigma_1$$

We aim for $10^i \sigma_1 \leq 0.0001$.

After a constant fraction of iterations, **the probability error becomes too high**.

After a constant fraction of iterations, **resample**!

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

How about $\mathrm{d}_v \leq n^{0.9}$?

How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

How about $\mathrm{d}_v \leq n^{0.9}$?

Assume that we simulate $\frac{\log n}{20}$ iterations.

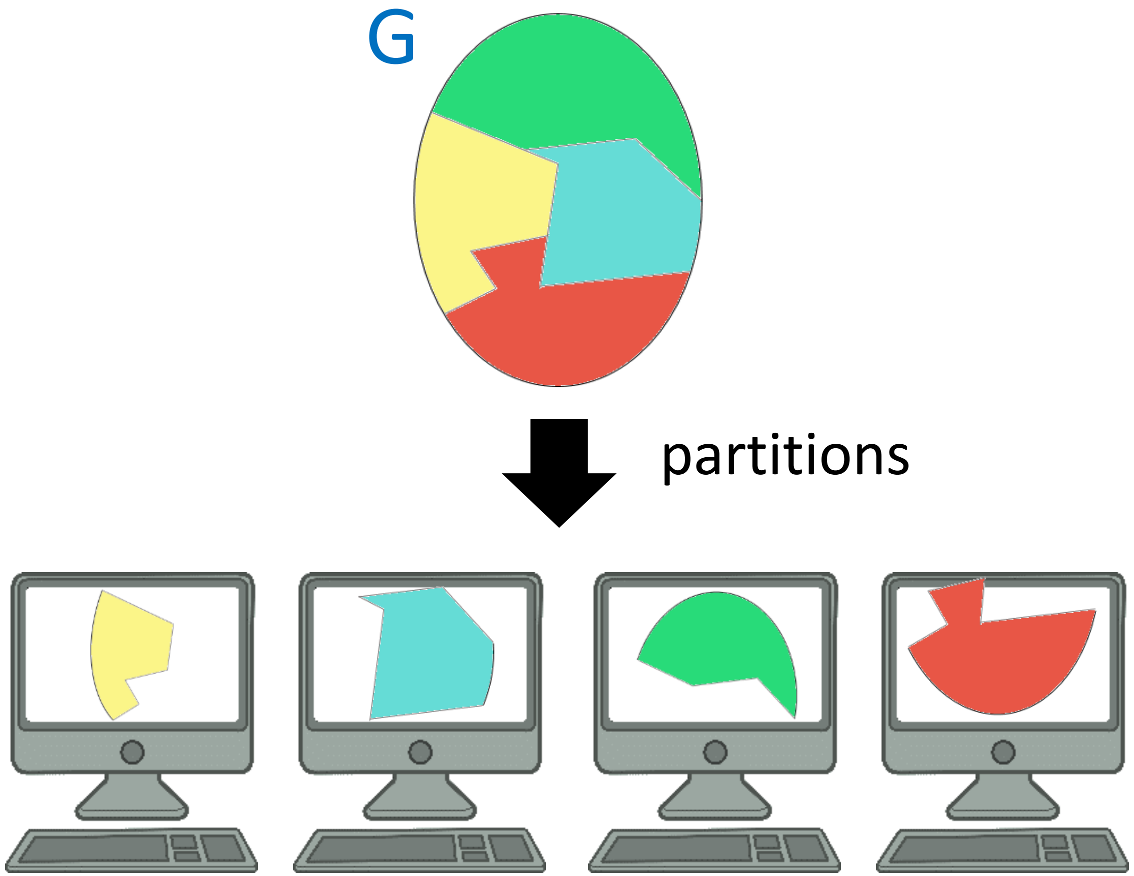
How much random thresholding gains?
I.e., what can we tell about $|y_v - \widetilde{y_v}|$?

How about $\mathrm{d}_v \leq n^{0.9}$?

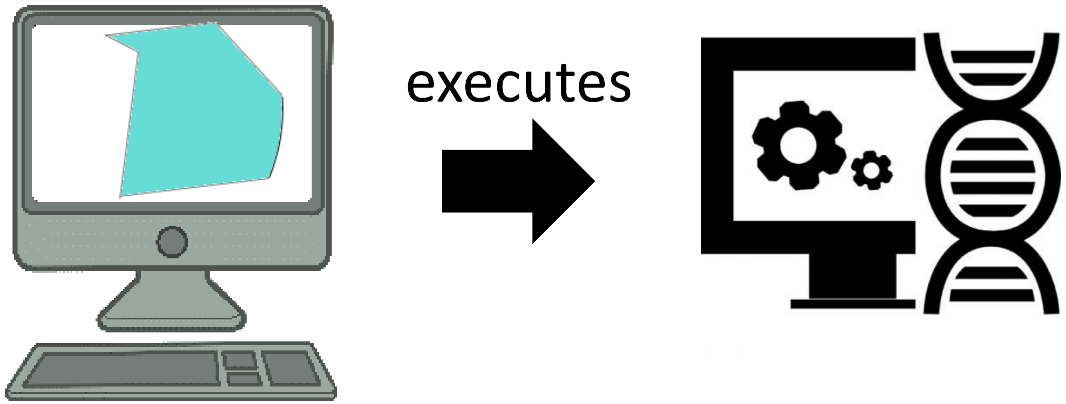
Assume that we simulate $\frac{\log n}{20}$ iterations.

Then, after the simulation, $x_e \leq \frac{n^{\frac{1}{20}}}{n} = \frac{1}{n^{0.95}}$

Hence, $\mathrm{y}_v \leq \mathrm{d}_v \, x_e \ll 1.$



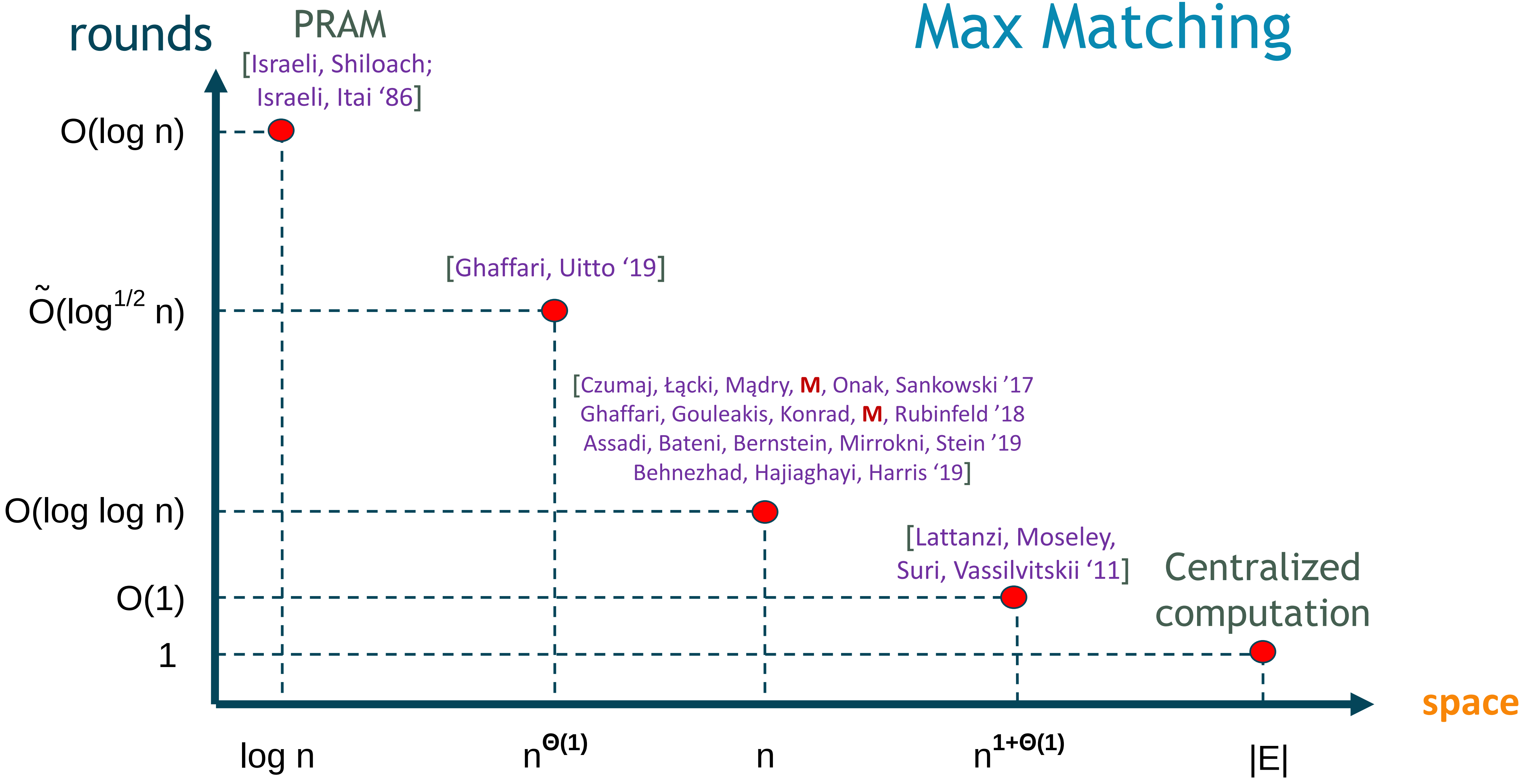
Random vertex partitioning

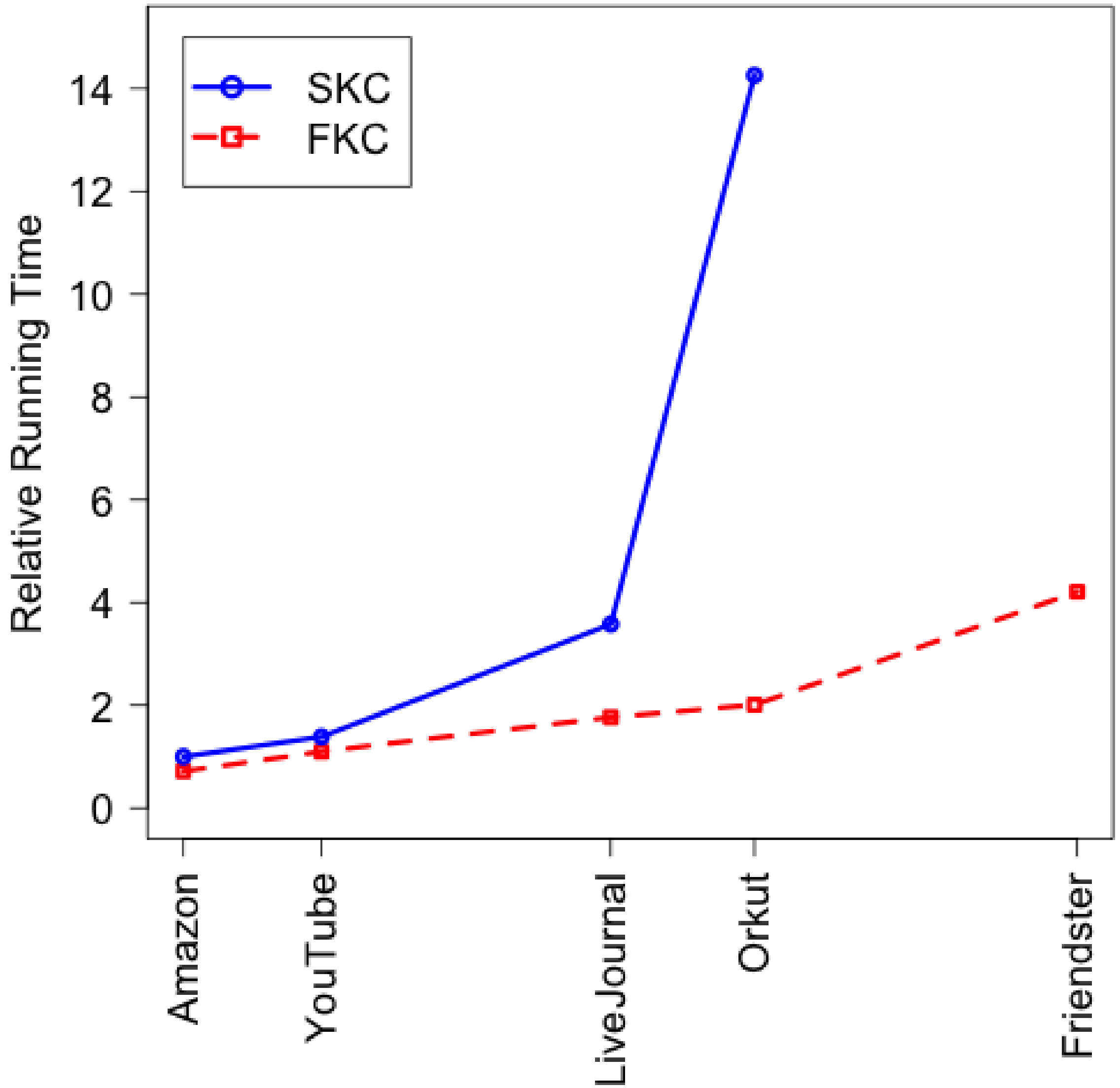


Simulation by randomly offsetting the threshold

Result: O(log n) → O(log log n) rounds n = |V|

Approximate Max Matching





[Ghaffari, Lattanzi, Mitrović, ICML '19]
(red line: our work; blue line: prior work)

Some open questions

1. $O(\log n)$ approximate set cover in $o(\log n)$ rounds with $O(n)$ space per machine.
2. $\Theta(1)$ approximate max matching in $o\left(\sqrt{\log n}\right)$ rounds with $O(n^{0.9})$ space per machine.
3. $\Theta(1)$ approximate densest subgraph in $o\left(\sqrt{\log n}\right)$ rounds with $O(n^{0.9})$ space per machine.
4. $\Theta(1)$ approximate densest subgraph in $\tilde{O}\left(\sqrt{\log n}\right)$ rounds with $O(n^{0.9})$ space per machine and $\tilde{O}(m)$ total space.