Decremental Matching in General Graphs

Sepehr Assadi* **Aaron Bernstein*** **Aditi Dudeja***

*Rutgers University

The Dynamic Graph Model

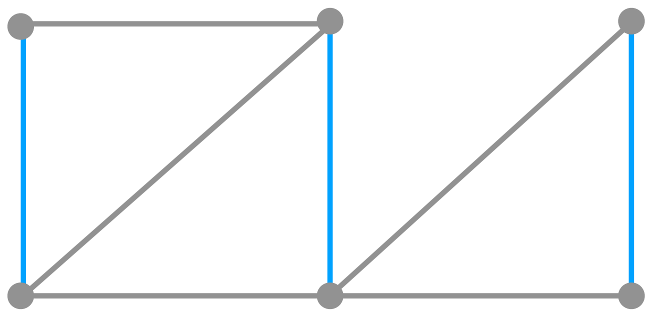
- Process a sequence of edge insertion and deletions.
- Maintain a large matching with a small update time (amortized/worst case).
- Incremental Model: Only edge insertions allowed.
- Decremental Model: Only edge deletions allowed.

The Dynamic Graph Model

- Process a sequence of edge insertion and deletions.
- Maintain a large matching with a small update time (amortized/worst case).
- Incremental Model: Only edge insertions allowed.
- Decremental Model: Only edge deletions allowed.

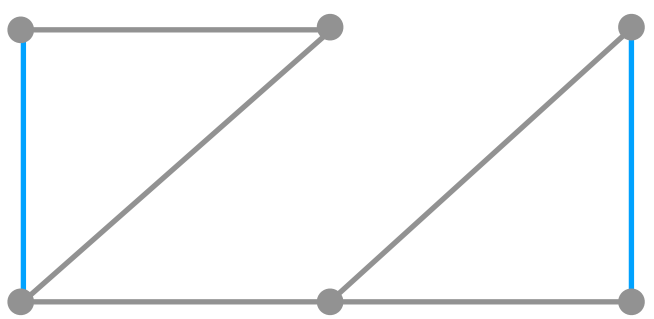
The Dynamic Graph Model

- Process a sequence of edge insertion and deletions.
- Maintain a large matching with a small update time (amortized/worst case).
- Incremental Model: Only edge insertions allowed.
- Decremental Model: Only edge deletions allowed.



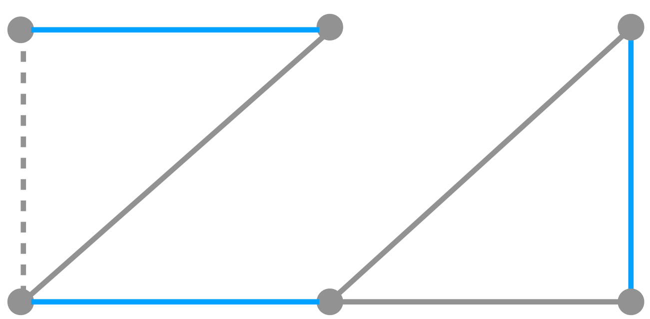
The Dynamic Graph Model

- Process a sequence of edge insertion and deletions.
- Maintain a large matching with a small update time (amortized/worst case).
- Incremental Model: Only edge insertions allowed.
- Decremental Model: Only edge deletions allowed.



The Dynamic Graph Model

- Process a sequence of edge insertion and deletions.
- Maintain a large matching with a small update time (amortized/worst case).
- Incremental Model: Only edge insertions allowed.
- Decremental Model: Only edge deletions allowed.



Background

Upper Bounds for $(1-\varepsilon)$ - approximation

	setting	update time	bipartite/general
[GP13]	fully dynamic	$O_\varepsilon(\sqrt{m})$	general
[GLSSS19]	incremental	$O_\varepsilon(1)$	general
[BGS20]	decremental	$O_\varepsilon(1)$	bipartite

Background

Upper Bounds for $(1-\varepsilon)$ - approximation

	setting	update time	bipartite/general
[GP13]	fully dynamic	$O_\varepsilon(\sqrt{m})$	general
[GLSSS19]	incremental	$O_\varepsilon(1)$	general
BGS20	decremental	$O_\varepsilon(1)$	bipartite

This Work: $(1-\varepsilon)$ - approximation in $O_\varepsilon(1)$ update time for general graphs.

Congestion Balancing [BGS20]

Let μ be the initial size of the matching. It is sufficient to solve the following problem in $\tilde{O}_{\varepsilon}(m)$ time:

1. Maintain a matching of size at least $\mu(1-\varepsilon)$ or,

2. Certify that maximum matching has dropped below $\mu(1-\varepsilon)$.

Congestion Balancing [BGS20]

Let μ be the initial size of the matching. It is sufficient to solve the following problem in $\tilde{O}_\varepsilon(m)$ time:

1. Maintain a matching of size at least $\mu(1-\varepsilon)$ or,

2. Certify that maximum matching has dropped below $\mu(1-\varepsilon)$.

Each time 2) happens start a new phase with new estimate $\mu(1-\varepsilon)$.

\# phases = $\log_{1+\varepsilon} n$

Congestion Balancing [BGS20]

- A lazy approach:

  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Congestion Balancing [BGS20]

- A lazy approach:

  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Runtime: $\Omega(n)$ amortized

Congestion Balancing [BGS20]

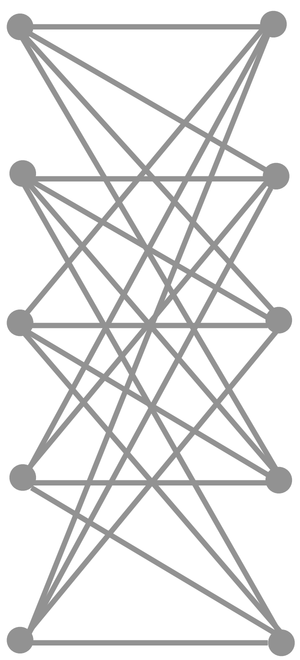
- A lazy approach:

  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Runtime: Ω(n) amortized



Congestion Balancing [BGS20]

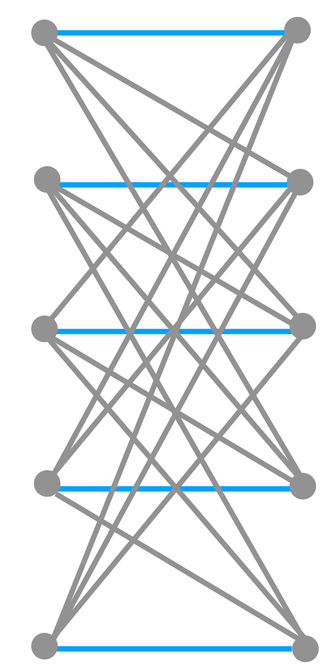
- A lazy approach:

1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

3. Recompute M.

Runtime: Ω(n) amortized



Phase 1

Congestion Balancing [BGS20]

- A lazy approach:

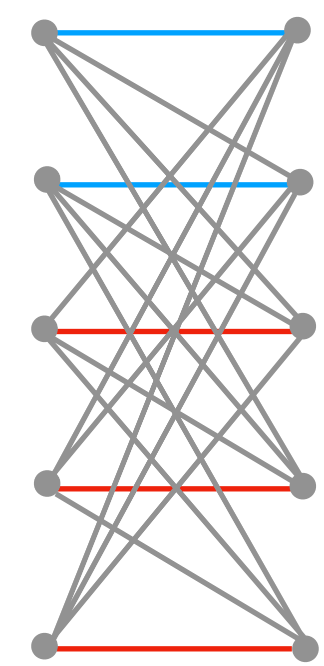
  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Runtime: $\Omega(n)$ amortized

Delete



Congestion Balancing [BGS20]

- A lazy approach:

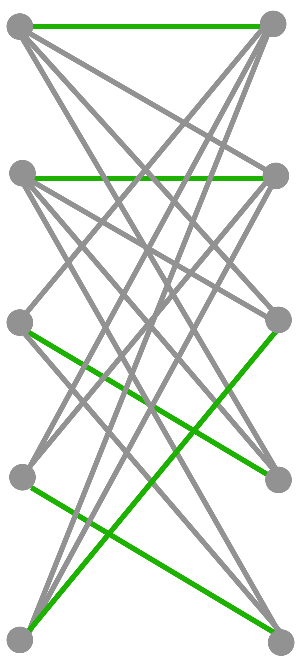
  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_{\varepsilon}(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Runtime: Ω(n) amortized

Phase 2



Congestion Balancing [BGS20]

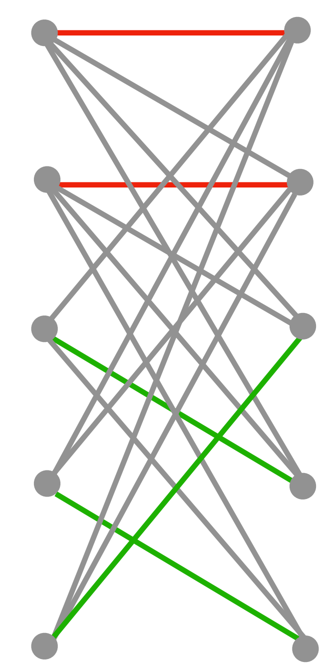
- A lazy approach:

  1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

  2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

  3. Recompute M.

Runtime: Ω(n) amortized



Delete

Congestion Balancing [BGS20]

- A lazy approach:

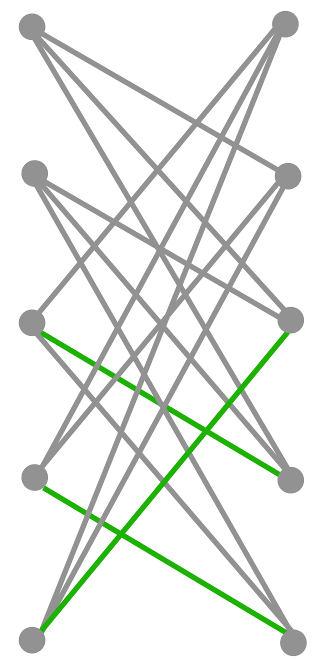
1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

3. Recompute M.

Runtime: Ω(n) amortized

Delete



Phases = Ω(n)

Congestion Balancing [BGS20]

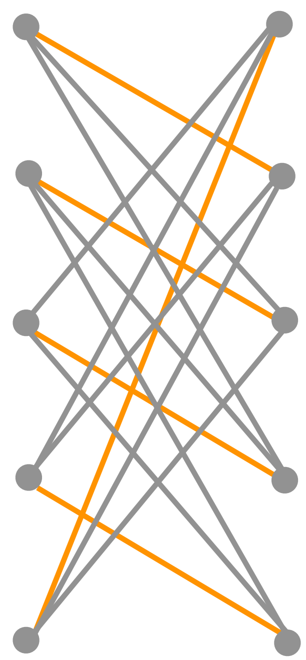
- A lazy approach:

1. Compute a $(1-\varepsilon)$- approximate maximum matching M in time $O_\varepsilon(m)$.

2. Do nothing until the adversary reduces |M| by a $(1-\varepsilon)$ factor.

3. Recompute M.

Runtime: Ω(n) amortized



Main Issue: Too much congestion on edges.

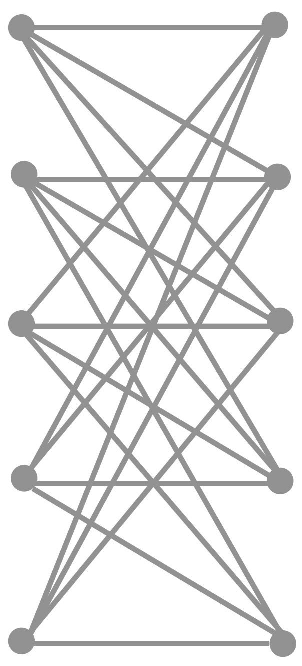
Congestion Balancing [BGS20]

Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]

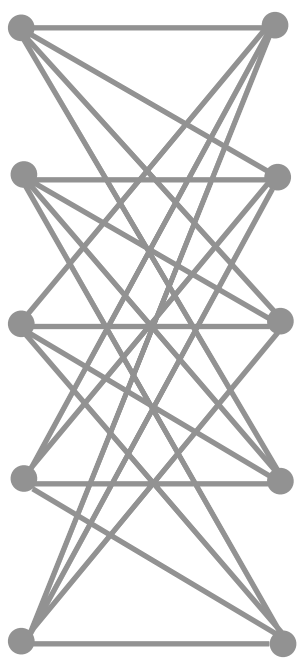
Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]



Congestion Balancing [BGS20]

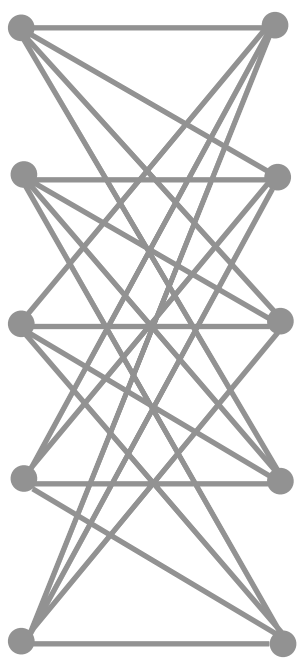
Solution: Enough to maintain a fractional matching. [W20, BK21]



Put flow $\frac{1}{n}$ on every edge.

Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]

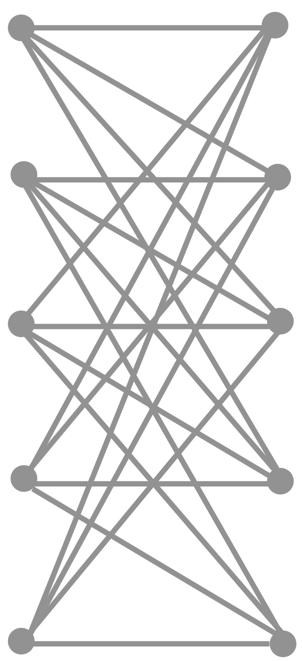


Put flow $\frac{1}{n}$ on every edge.

Adversary has to delete $\Omega(n^2)$ edges.

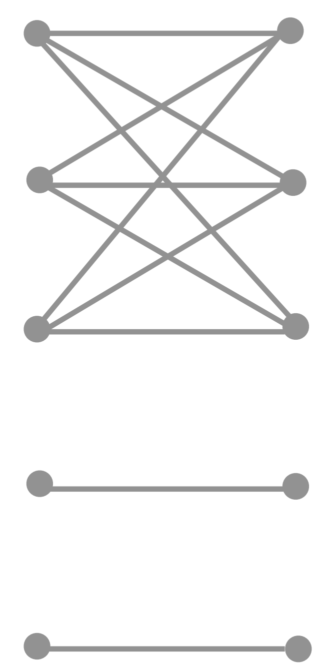
Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]



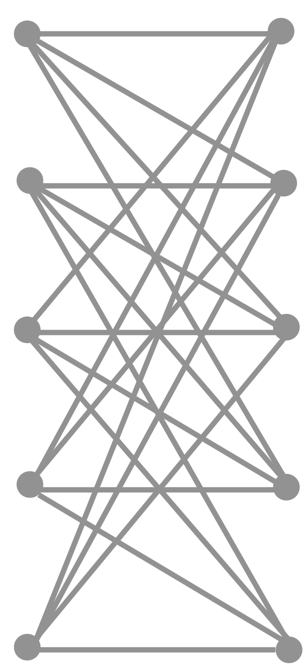
Put flow $\frac{1}{n}$ on every edge.

Adversary has to delete $\Omega(n^2)$ edges.



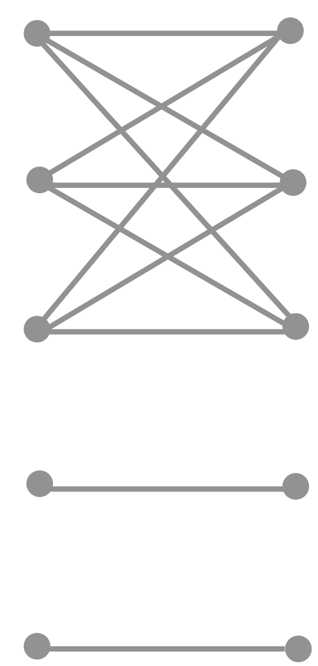
Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]



Put flow $\frac{1}{n}$ on every edge.

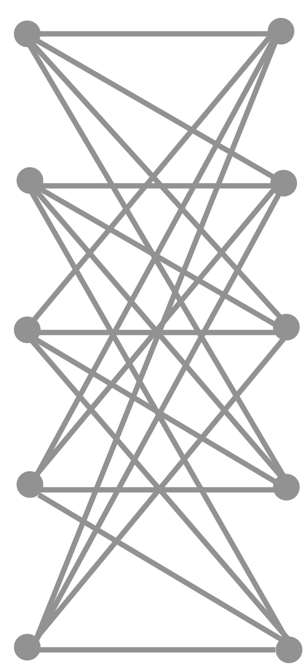
Adversary has to delete $\Omega(n^2)$ edges.



May have to put large flow on crucial edges.

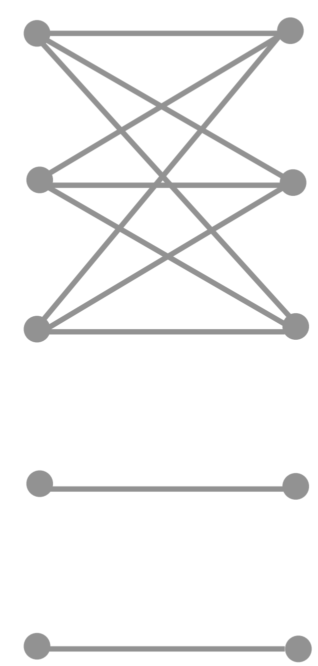
Congestion Balancing [BGS20]

Solution: Enough to maintain a fractional matching. [W20, BK21]



Put flow $\frac{1}{n}$ on every edge.

Adversary has to delete $\Omega(n^2)$ edges.



May have to put large flow on crucial edges.

Adversary can’t delete too many of them.

Congestion Balancing [BGS20]

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Congestion Balancing [BGS20]

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

    - Estimate matching size. If smaller than $(1 - \epsilon)\mu$ then terminate.

    - Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
    - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Congestion Balancing [BGS20]

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

    - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

    - Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
    - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Congestion Balancing [BGS20]

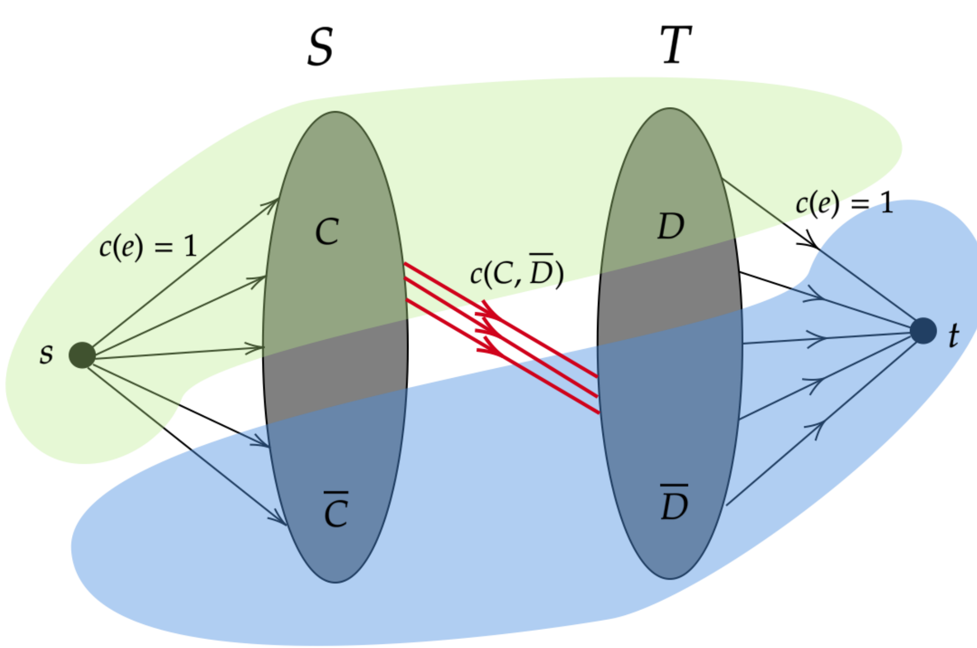
1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

- Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

- Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
- If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**



Congestion Balancing [BGS20]

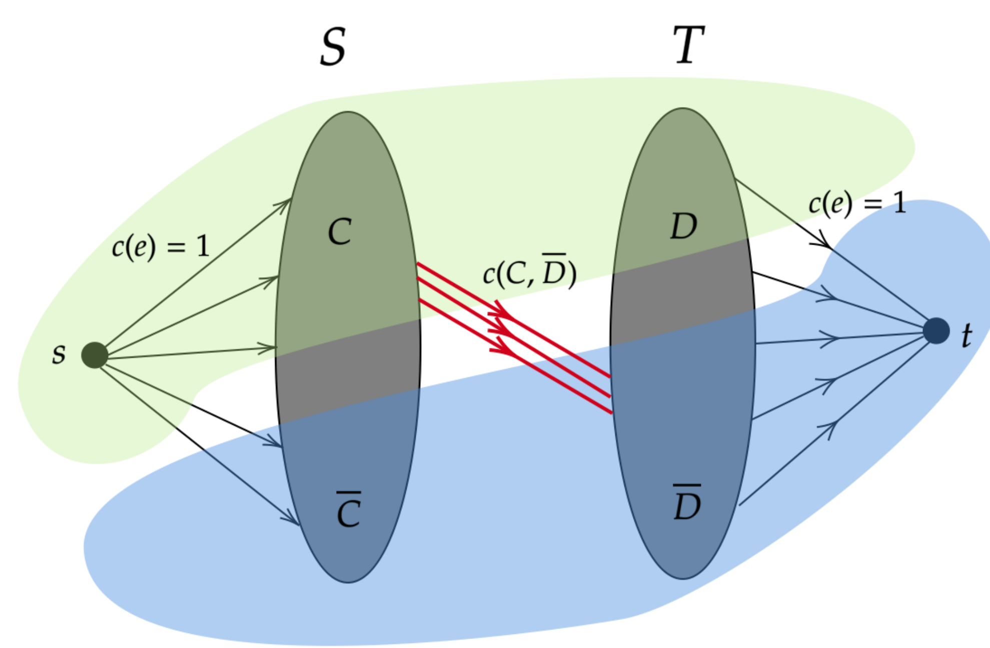
1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

- Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

- Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
- If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**



- Compute fractional matching using max-flow.
- Bottleneck edges correspond to min-cut.

Congestion Balancing [BGS20]

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

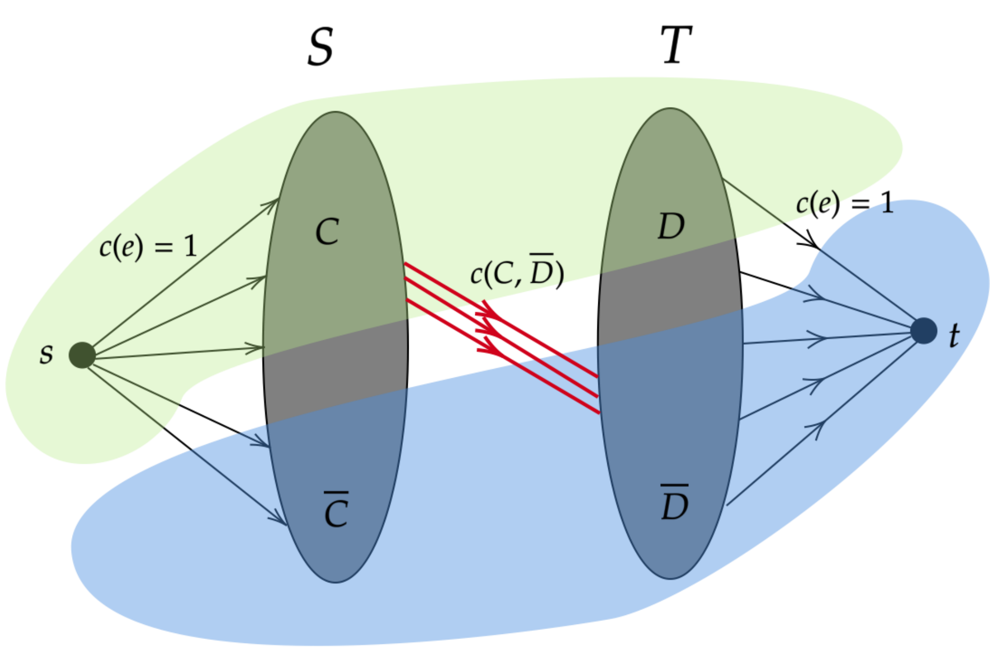
2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute fractional matching obeying capacities $\{c(e)\}_{e \in E}$.
   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Runtime: $\tilde{O}(m)$

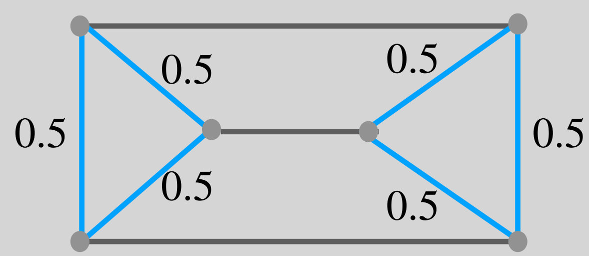


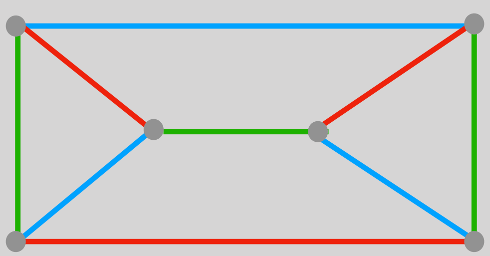
- Compute fractional matching using max-flow.
- Bottleneck edges correspond to min-cut.

Road Blocks for General Graphs

Road Blocks for General Graphs

Not every fractional matching is good.

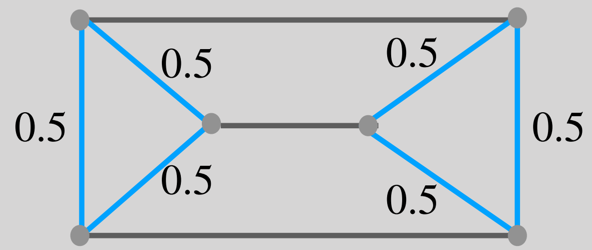


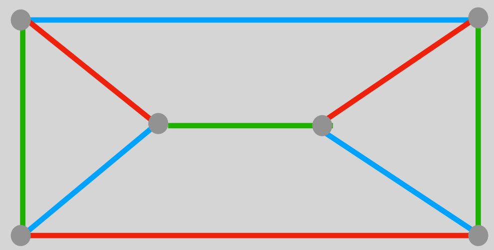


Road Blocks for General Graphs

- Doesn't obey blossom constraints!
- Converse also true.

Not every fractional matching is good.

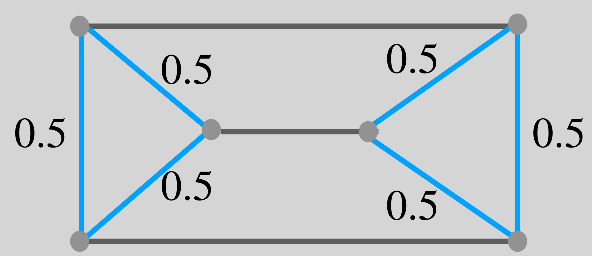


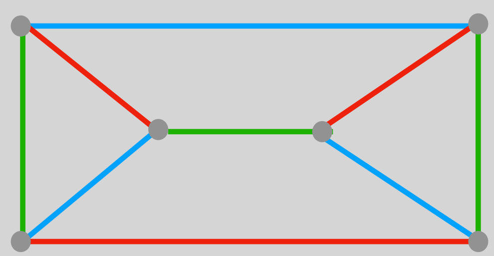


Road Blocks for General Graphs

- Doesn't obey blossom constraints!
- Converse also true.

Not every fractional matching is good.





How to determine bottleneck edges?

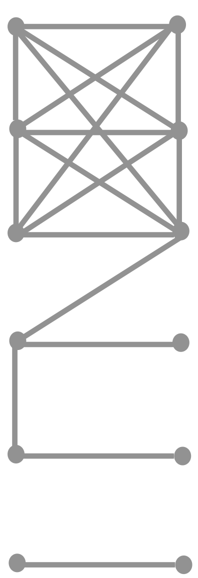
Addressing Road Block 1

Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

Addressing Road Block 1

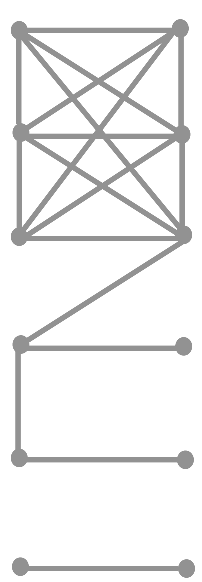
Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.



$$c(e) = \frac{1}{n} \forall e \in E$$

Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

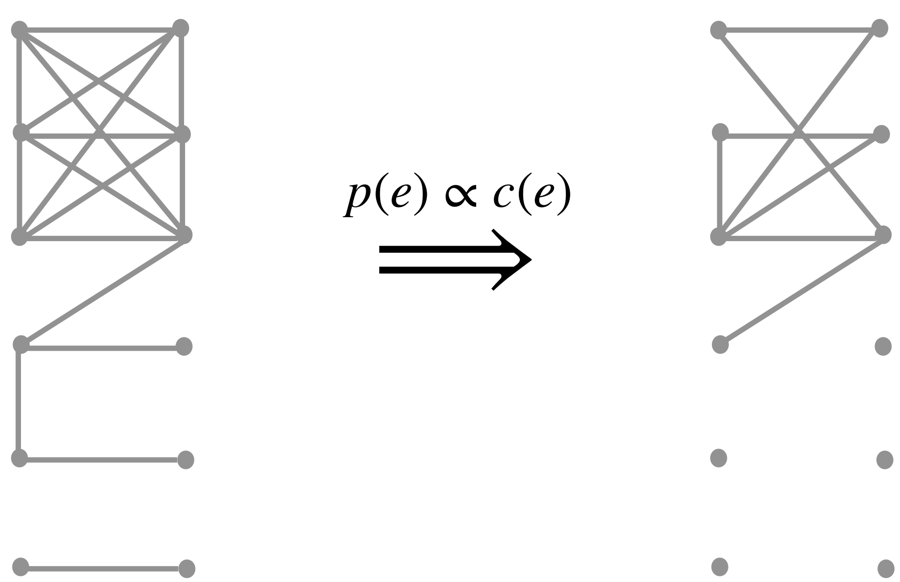


$$c(e) = \frac{1}{n} \forall e \in E$$

$$\xRightarrow{p(e) \propto c(e)}$$

Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

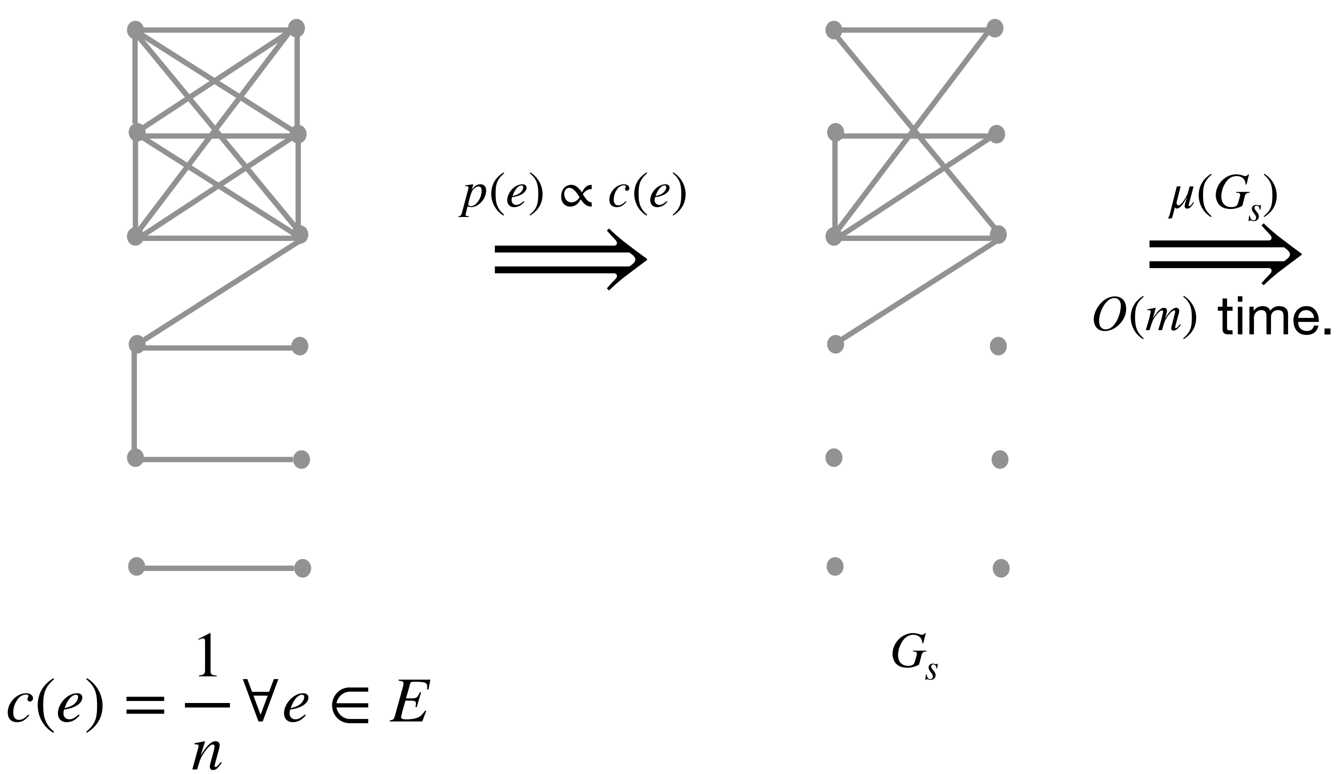


$c(e) = \frac{1}{n} \forall e \in E$

G_s

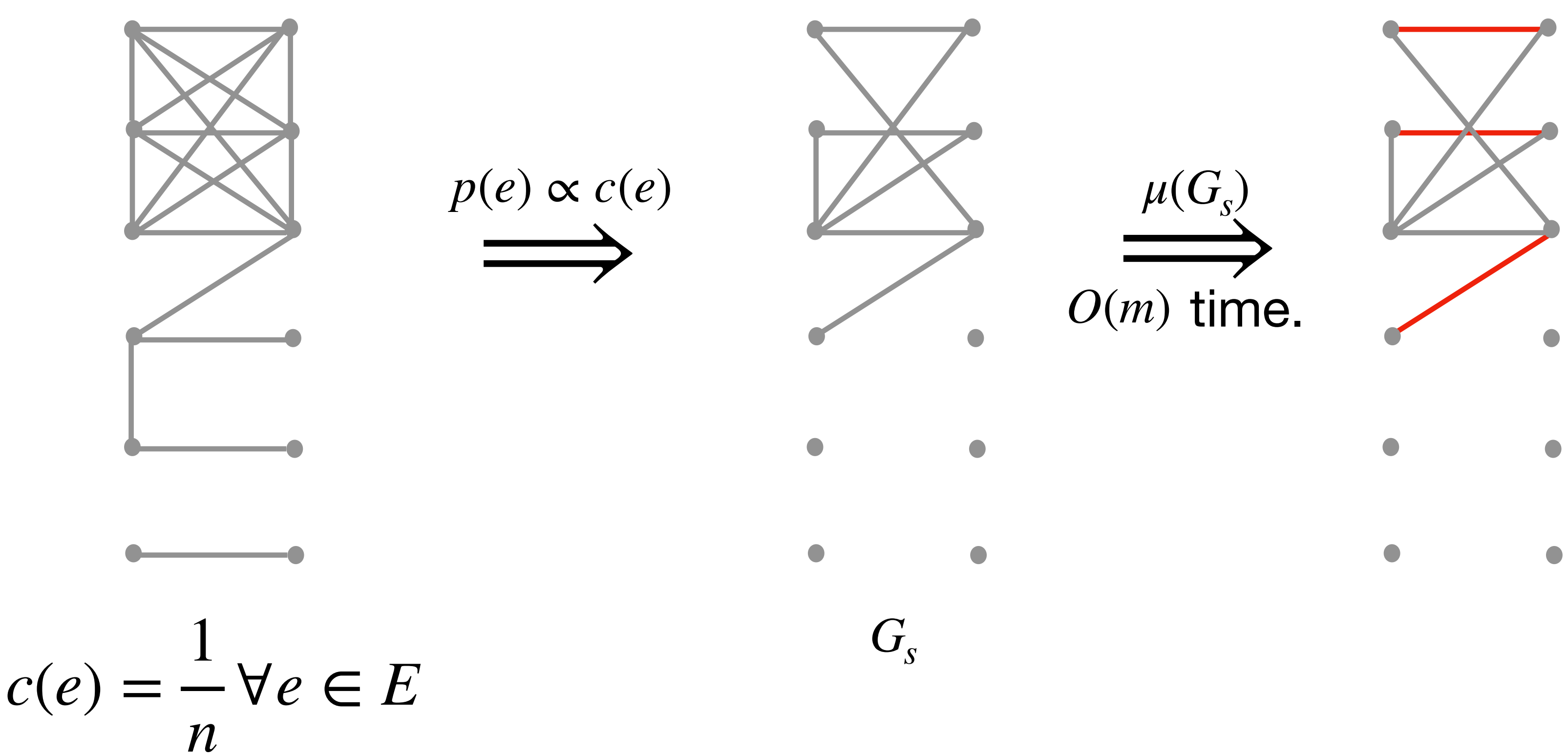
Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.



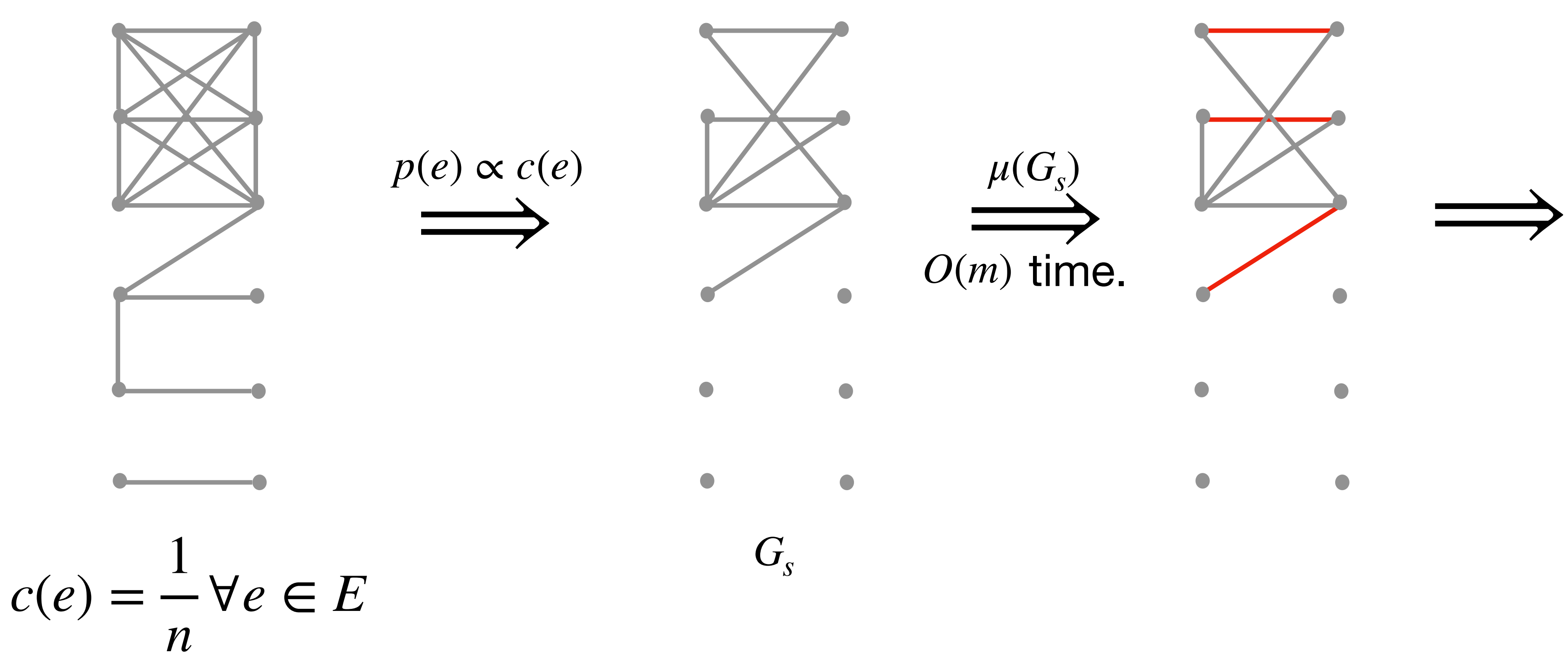
Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.



Addressing Road Block 1

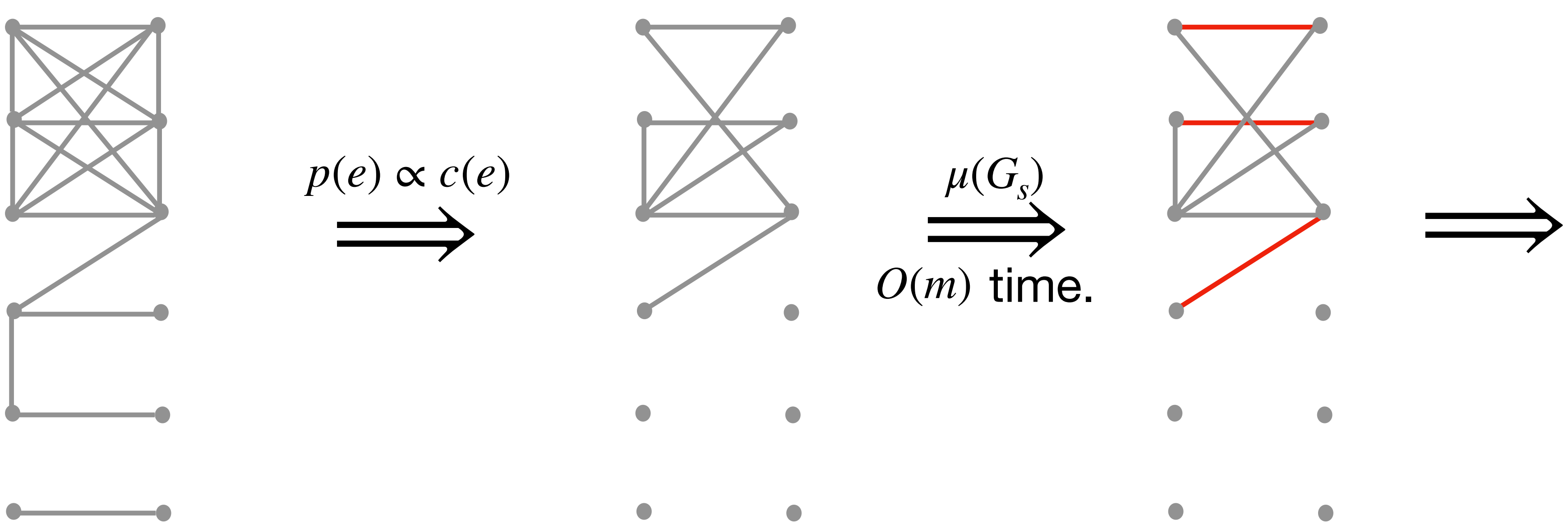
Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.



Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

Else $\mu(G_s)$ is large and good fractional matching exists. Find it!

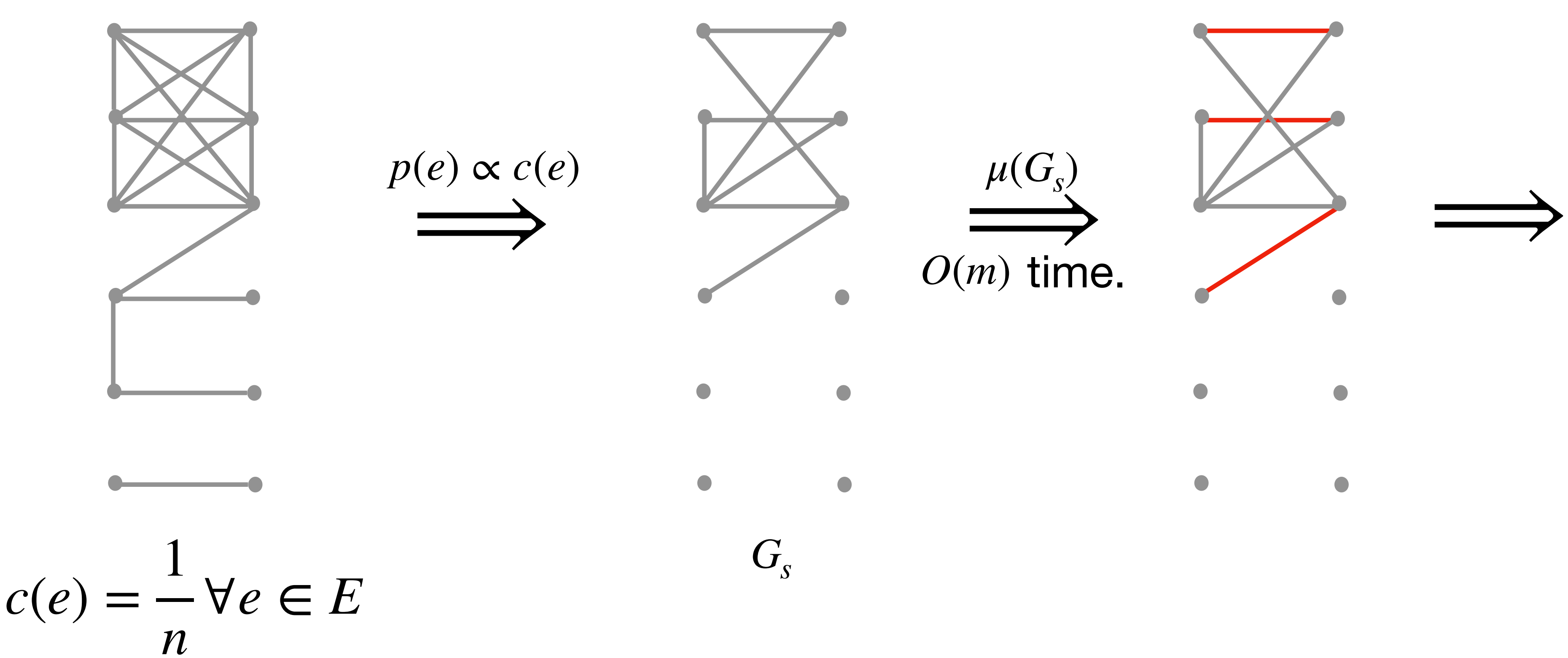


$$c(e) = \frac{1}{n} \forall e \in E$$

G_s

Addressing Road Block 1

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

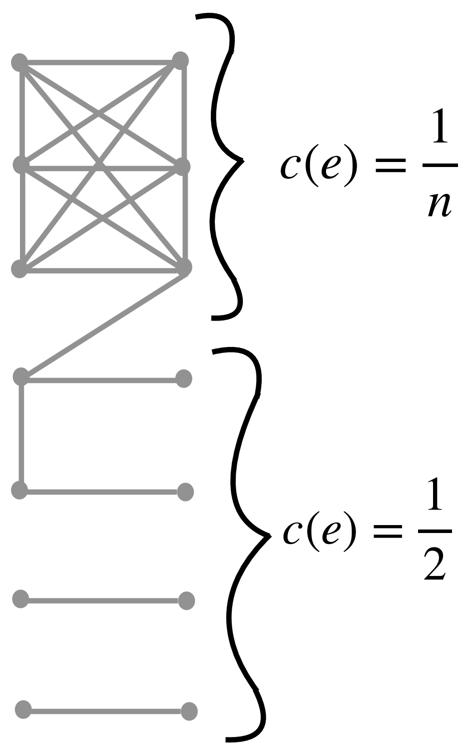


Else $\mu(G_s)$ is large and good fractional matching exists. Find it!

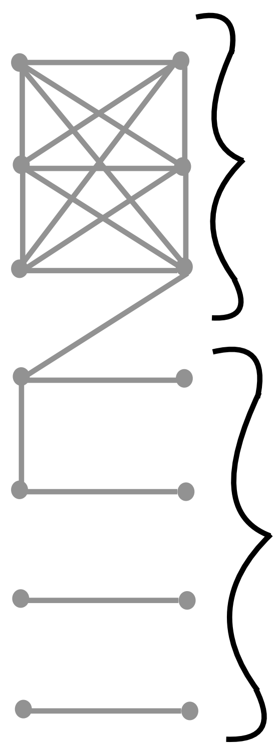
If $\mu(G_s)$ is small, then $\mu(G, c)$ is small. Increase capacity.

When $\mu(G_s)$ is large: Finding a good fractional matching

When $\mu(G_s)$ is large: Finding a good fractional matching



When $\mu(G_s)$ is large: Finding a good fractional matching

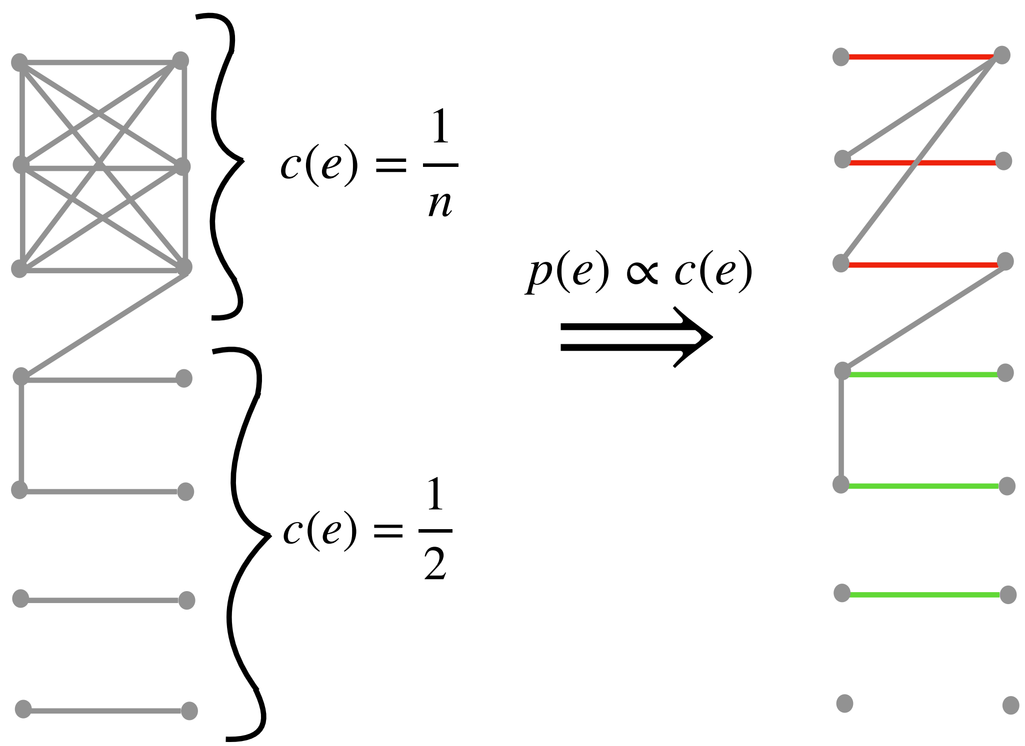


$c(e) = \frac{1}{n}$

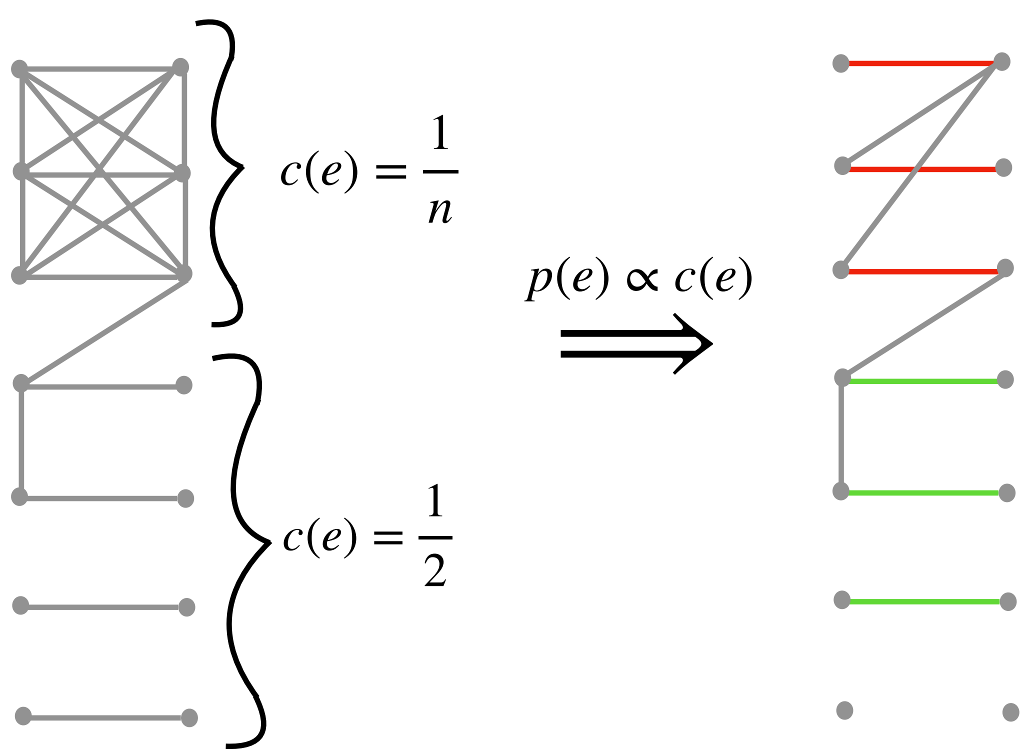
$c(e) = \frac{1}{2}$

$$\xRightarrow{p(e) \propto c(e)}$$

When $\mu(G_s)$ is large: Finding a good fractional matching

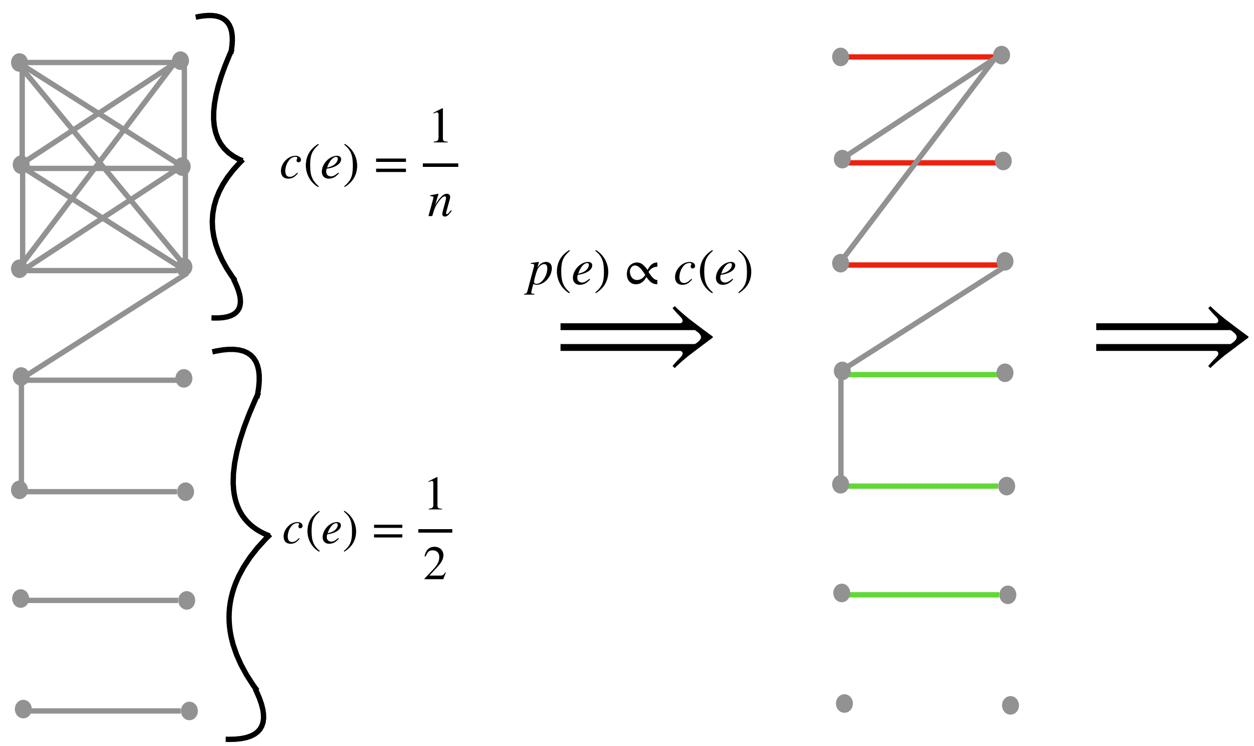


When $\mu(G_s)$ is large: Finding a good fractional matching



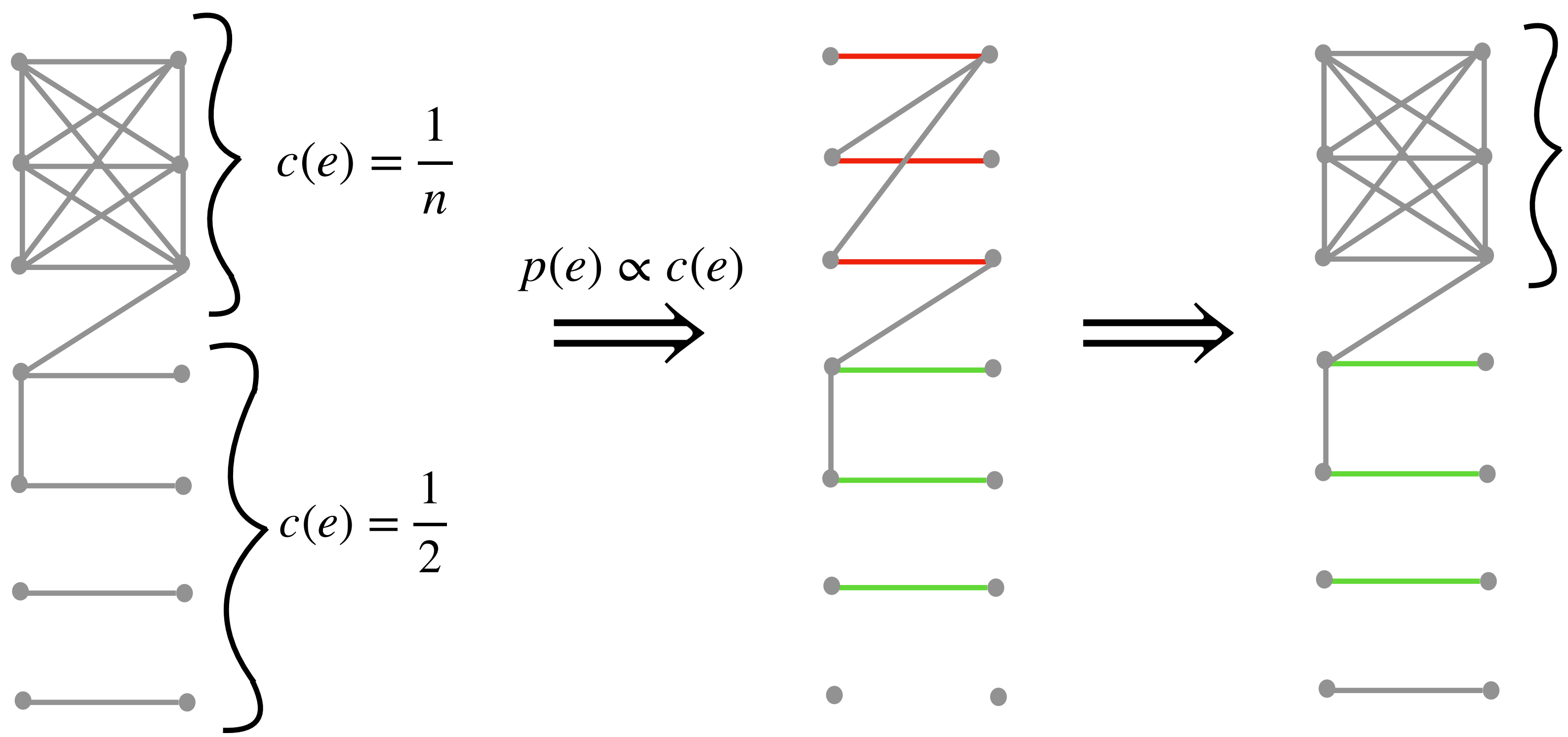
$\mu(G_s)$ is large.

When $\mu(G_s)$ is large: Finding a good fractional matching



$\mu(G_s)$ is large.

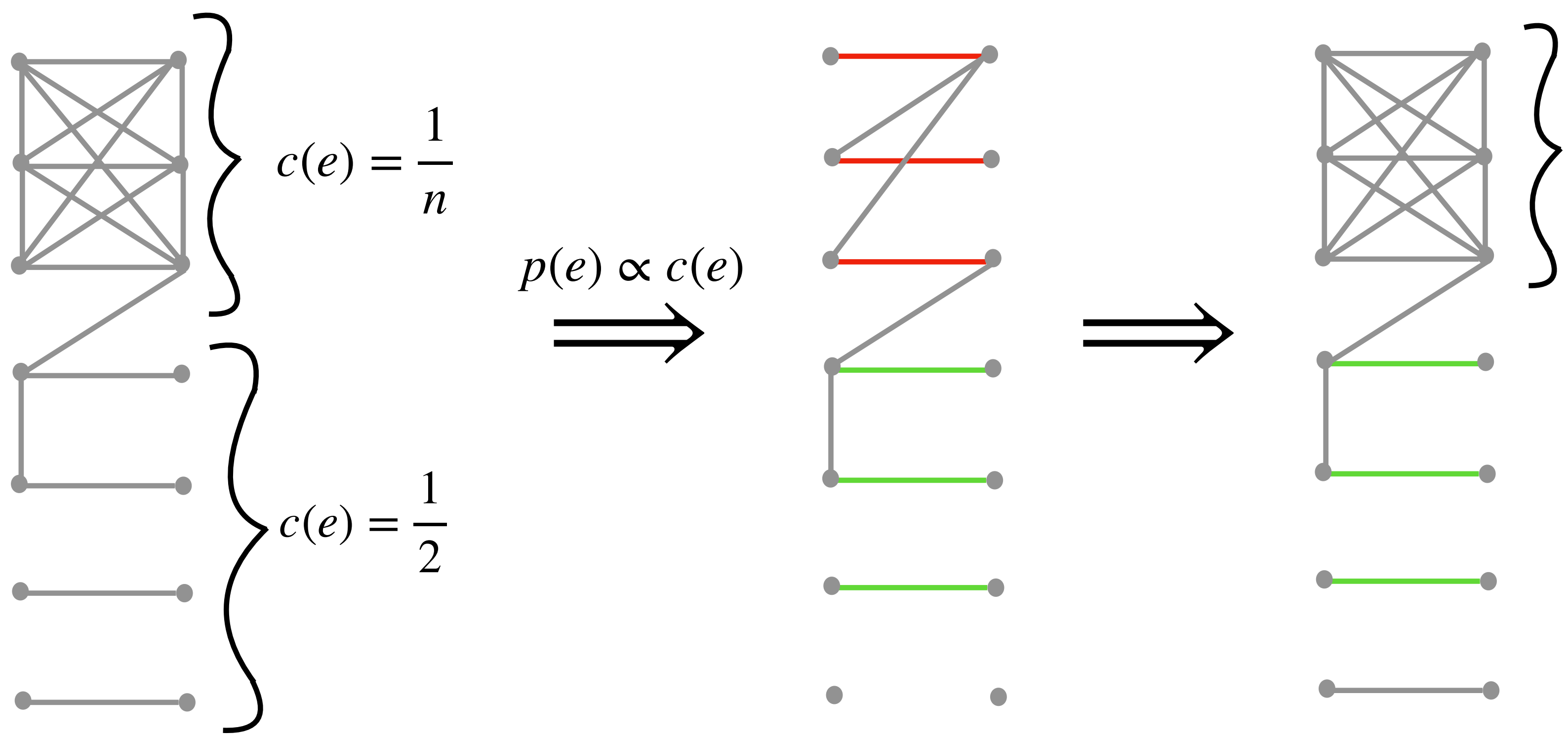
When $\mu(G_s)$ is large: Finding a good fractional matching



$\mu(G_s)$ is large.

Find a fractional matching f using max flow.

When $\mu(G_s)$ is large: Finding a good fractional matching



$\mu(G_s)$ is large.

Find a fractional matching f using max flow.

Theorem 2: f + green edges has value $\approx \mu(G_s)$ and satisfies blossom constraints.

Addressing Roadblock 2

Addressing Roadblock 2

Known: There is an $O_\varepsilon(m)$ time algorithm that computes $(1-\varepsilon)$ approximate maximum matching. Moreover, the algorithm also solves dual problem “approximately”.

Addressing Roadblock 2

Known: There is an $O_\varepsilon(m)$ time algorithm that computes $(1-\varepsilon)$ approximate maximum matching. Moreover, the algorithm also solves dual problem “approximately”.

Theorem 3: Suppose we solve the dual problem on G_s, then using that solution, we can determine bottleneck edges.

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 1: Let G_s be uncapacitated graph obtained by sampling every edge e with probability $p(e) \propto c(e)$, then $\mu(G_s) \approx \mu(G, c)$.

Putting Things Together

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 2: $f +$ green edges has value $\approx \mu(G_s)$ and satisfies blossom constraints.

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

    - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

    - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

    - If value is large, compute such a fractional matching.

    - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 2: $f +$ green edges has value $\approx \mu(G_s)$ and satisfies blossom constraints.

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

    - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

    - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

    - If value is large, compute such a fractional matching.

    - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

- Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

- Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

- If value is large, compute such a fractional matching.

- If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 3: Suppose we solve the dual problem on G_s, then using that solution, we can determine bottleneck edges.

Putting Things Together

1. Initially, set $c(e) = \frac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

   - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

   - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

   - If value is large, compute such a fractional matching.

   - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Theorem 3: Suppose we solve the dual problem on G_s, then using that solution, we can determine bottleneck edges.

Putting Things Together

1. Initially, set $c(e) = \dfrac{1}{n^2} \forall e \in E$.

2. **Initial phase:**

    - Estimate matching size. If smaller than $(1-\epsilon)\mu$ then terminate.

    - Compute value of fractional matching obeying capacities $\{c(e)\}_{e \in E}$ and blossom constraints.

    - If value is large, compute such a fractional matching.

    - If the matching is too small, then increase capacity along crucial edges until fractional matching is large enough.

3. **Process deletions.**

Open Questions

1. Can the algorithm be derandomized?

2. Can we improve dependence on $\frac{1}{\varepsilon}$?