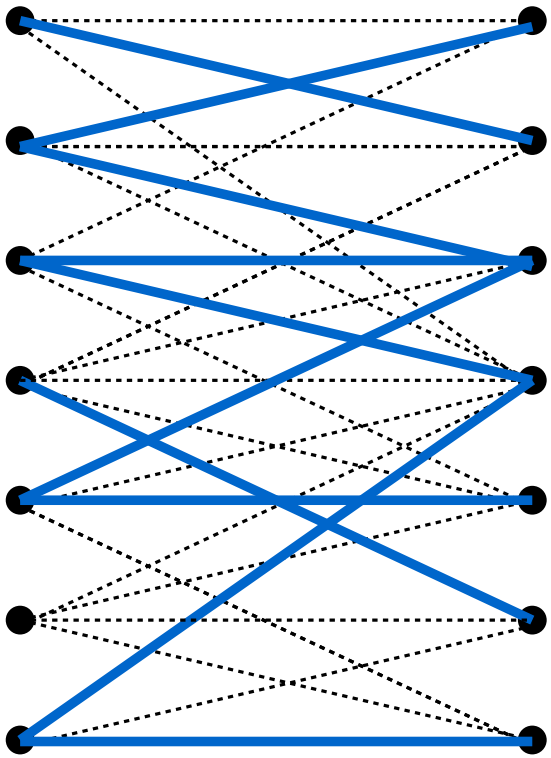
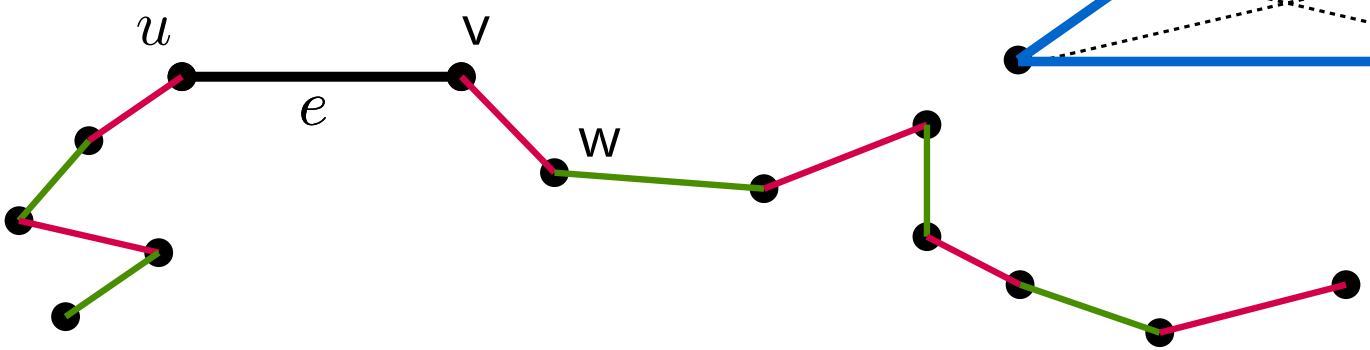
Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time

Joakim Blikstad[†] & Peter Kiss[*]

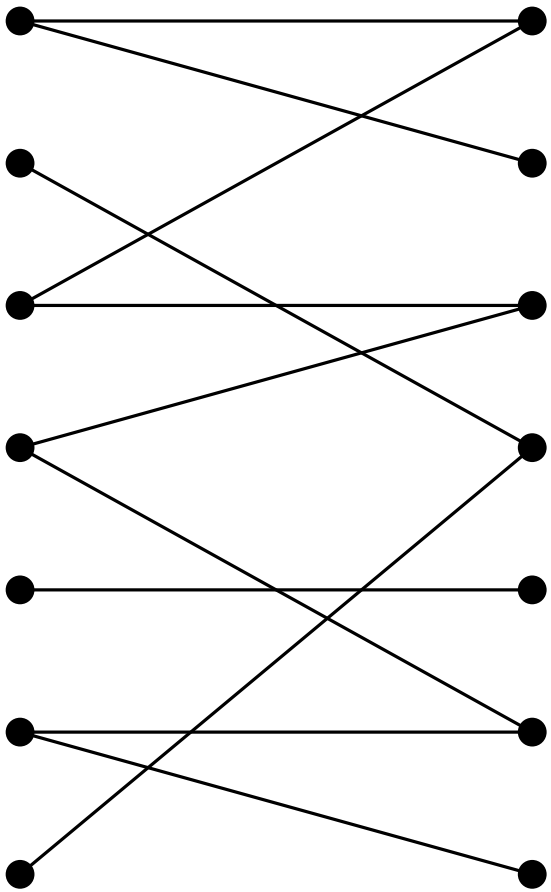
ESA 2023

[†]KTH Royal Institute of Technology
[*]University of Warwick
[†*]Max-Planck-Institut für Informatik





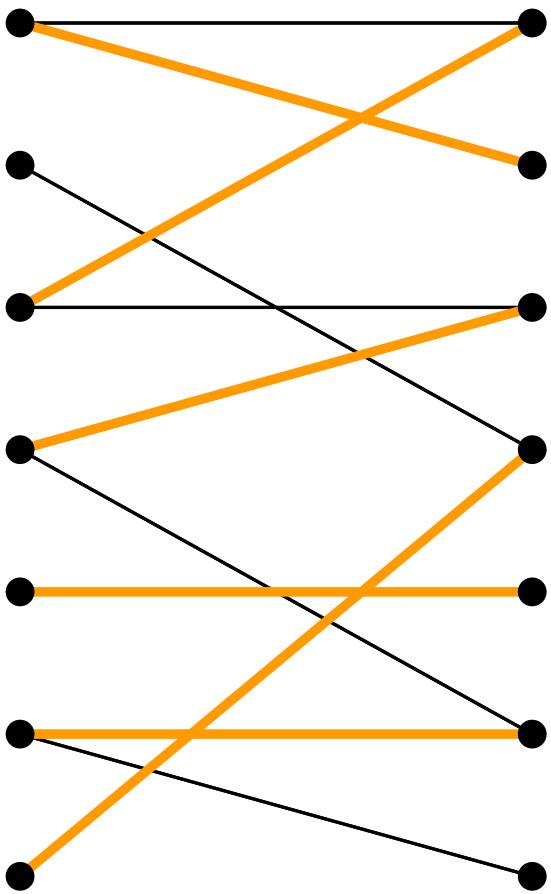
Bipartite Matching



Bipartite Graph $G = (V, E)$

- $n = |V|$, $m = |E|$

Bipartite Matching



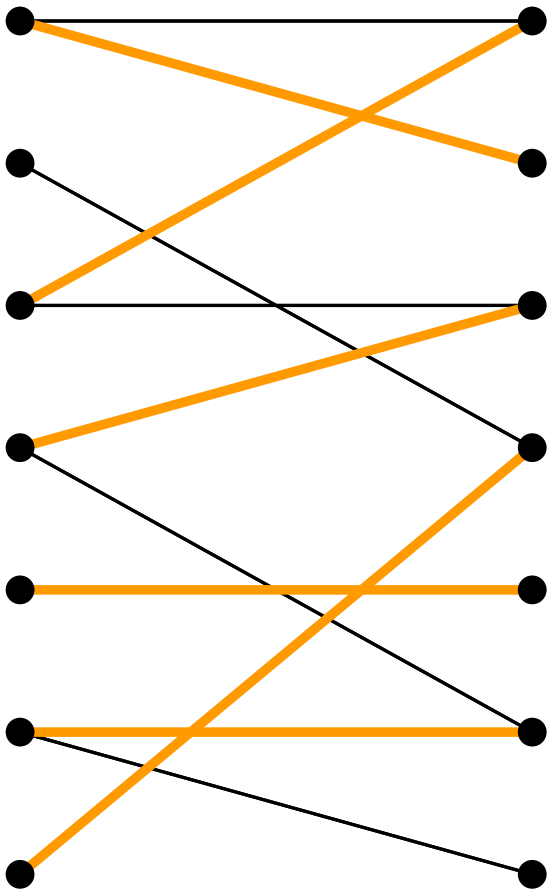
Bipartite Graph $G = (V, E)$

- $n = |V|$, $m = |E|$

Matching $M \subseteq E$

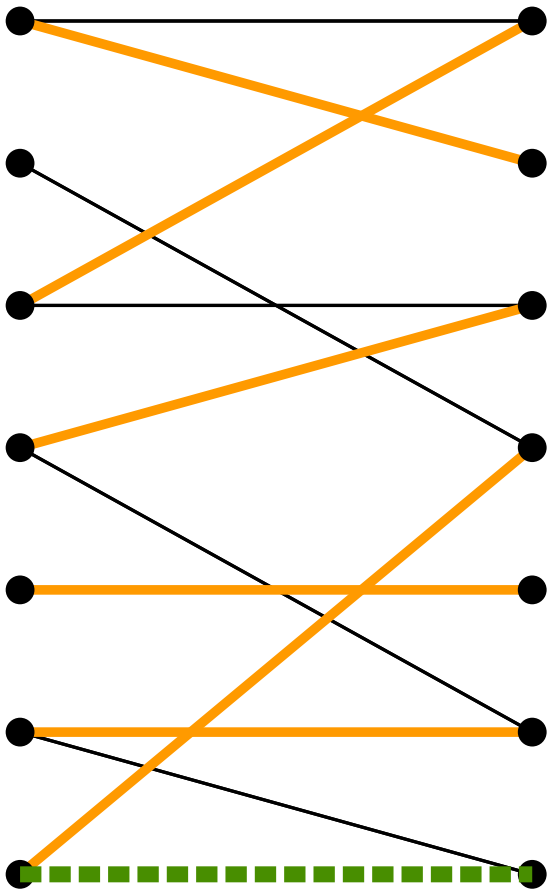
- share no vertices
- $\mu(G) = |\text{maximum matching}|$

Dynamic Model



Graph G changes over time

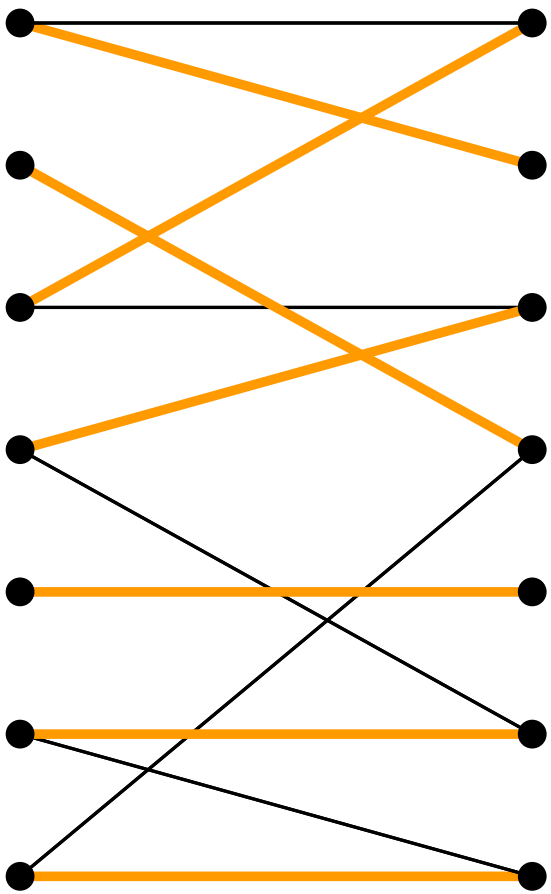
Dynamic Model



Graph G changes over time

- Insertions

Dynamic Model

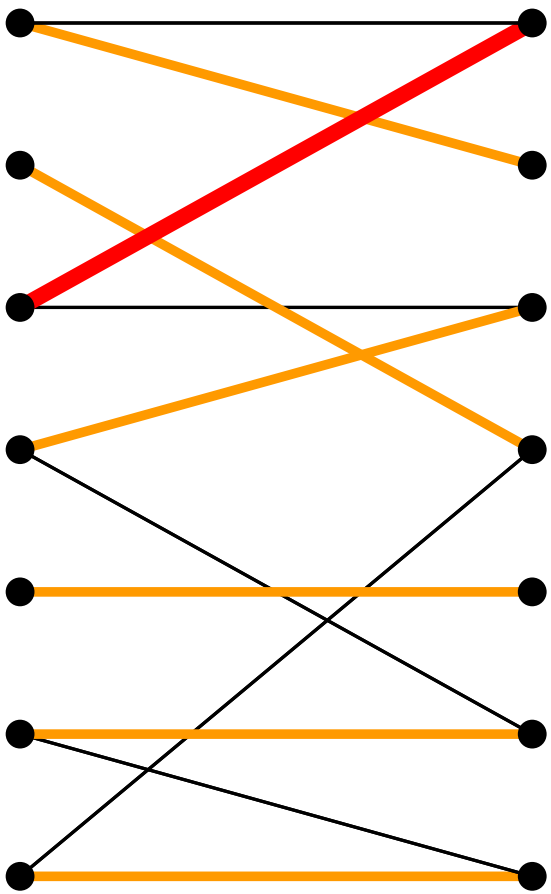


Graph G changes over time

- Insertions

Goal: Maintain Maximum Matching

Dynamic Model

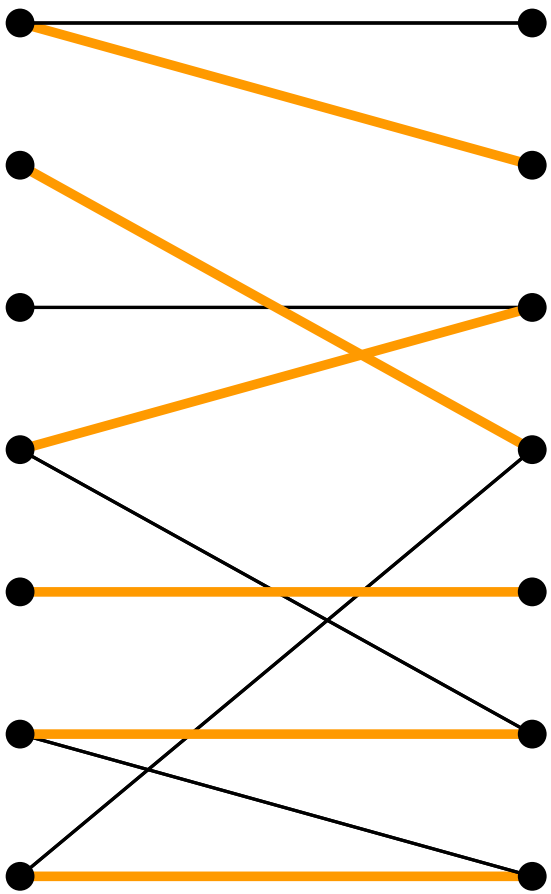


Graph G changes over time

- Insertions
- Deletions (not in this talk)

Goal: Maintain Maximum Matching

Dynamic Model



Graph G changes over time

- Insertions
- Deletions (not in this talk)

Goal: Maintain Maximum Matching

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx [Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? — $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx [Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES — $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? — $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES — $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Even faster: Constant time (independent of n)?

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES — $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Even faster: Constant time (independent of n)?

YES — $1/\varepsilon^{O(1/\varepsilon)}$ update time (incremental)
[Grandoni-Leonardi-Sankowski-Schwiegelshohn-Solomon SODA'19]

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES — $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Even faster: Constant time (independent of n)?

YES — $1/\varepsilon^{O(1/\varepsilon)}$ update time (incremental)
[Grandoni-Leonardi-Sankowski-Schwiegelshohn-Solomon SODA'19]

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES — $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Even faster: Constant time (independent of n)?

YES — $1/\varepsilon^{O(1/\varepsilon)}$ update time (incremental) [Grandoni-Leonardi-Sankowski-Schwiegelshohn-Solomon SODA'19]

Even faster: Best of both?

Dynamic Matching — State of the Art

Can we maintain maximum matching efficiently?

NO — Need $\tilde{\Omega}(n)$ update time [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

What about $(1-\varepsilon)$-approximate maximum matching?

Maybe? $n^{1-\Omega_\varepsilon(1)}$ update time [Bhattacharya-Kiss-Saranurak '23]

$O(\text{polylog}(n))$ update time for $\frac{1}{2}$-approx or 0.585-approx
[Baswana-Gupta-Sen FOCS'11] [Behnezhad-Khanna SODA'23] [Bhattacharya-Kiss-Saranurak-Wajc SODA'23]

Incremental/Decremental $(1-\varepsilon)$-approx maximum matching?

YES $O(\text{polylog}(n)/\text{poly}(\varepsilon))$ update time [Gupta FSTTCS'14] [Bhattacharya-Kiss-Saranurak SODA'23] [Jambulapati-Jin-Sidford-Tian ICALP'22] [Bernstein-Probst Gutenberg-Saranurak FOCS'20]

Even faster: Constant time (independent of n)?

YES $1/\varepsilon^{O(1/\varepsilon)}$ update time (incremental)
[Grandoni-Leonardi-Sankowski-Schwiegelshohn-Solomon SODA'19]

Even faster: Best of both?

YES — Our Result

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

- $O(n/\varepsilon^6 + m/\varepsilon^5)$ total update time

  ($O(1/\varepsilon^6)$ per update, amortized)

- Edge Insertions

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

- $O(n/\varepsilon^6 + m/\varepsilon^5)$ total update time

  ($O(1/\varepsilon^6)$ per update, amortized)

- Edge Insertions
- (+ Vertex Deletions)

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

- $O(n/\varepsilon^6 + m/\varepsilon^5)$ total update time

  ($O(1/\varepsilon^6)$ per update, amortized)

- Edge Insertions
- (+ Vertex Deletions)
- Only for bipartite graphs

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

- $O(n/\varepsilon^6 + m/\varepsilon^5)$ total update time

  ($O(1/\varepsilon^6)$ per update, amortized)

- Edge Insertions
- (+ Vertex Deletions)
- Only for bipartite graphs
- *Fractional* maximum matching in *non*-bipartite graphs

Our Results

"Incremental $(1-\epsilon)$-approximate Dynamic Matching in $O(\mathrm{poly}(1/\epsilon))$ Update Time"

- $O(n/\varepsilon^6 + m/\varepsilon^5)$ total update time

  ($O(1/\varepsilon^6)$ per update, amortized)

- Edge Insertions

- (+ Vertex Deletions)

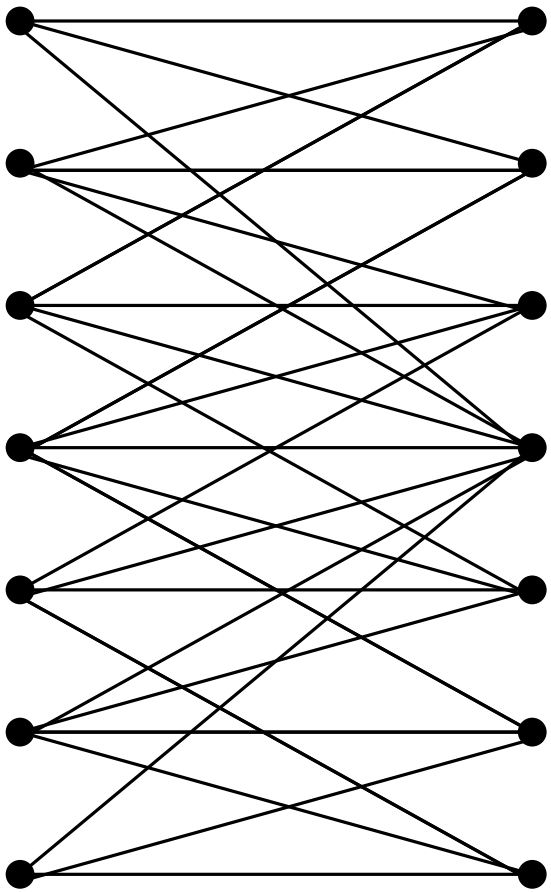
- Only for bipartite graphs

- *Fractional* maximum matching in *non*-bipartite graphs

Bonus: It's quite simple :)

Our Techniques

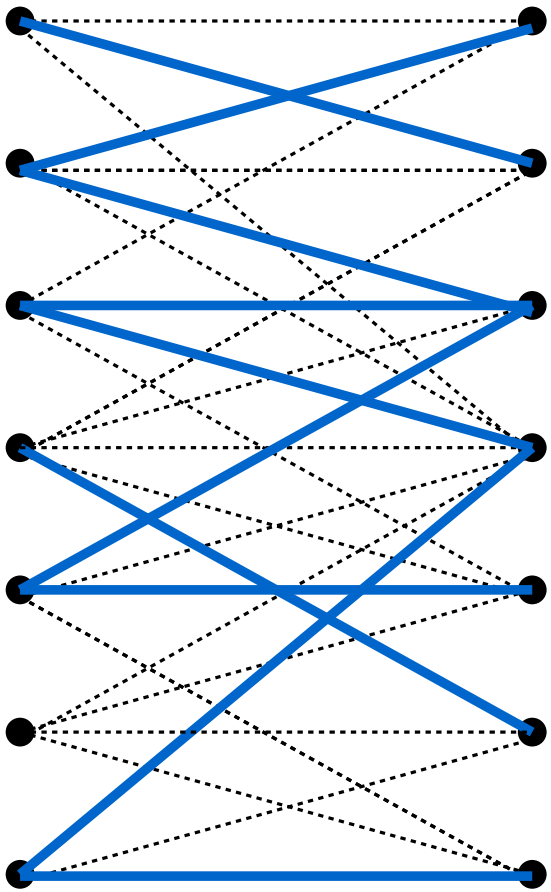
($\mu(G) = |\text{maximum matching}|$)



Graph G has many edges. . .

Our Techniques

($\mu(G) = |\text{maximum matching}|$)



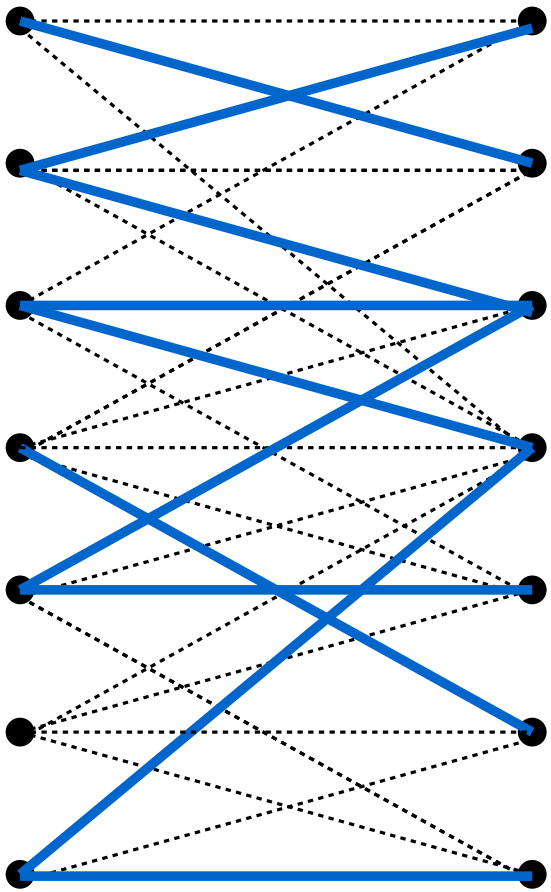
Graph G has many edges...

Keep track of a subgraph H:

- "EDCS" [Bernstein-Stein ICALP'15]
- Sparse: $|E(H)| \leq \mu(G)/\varepsilon^2$
- Preserves good matching: $\mu(H) \geq (1-\varepsilon)\mu(G)$
- Efficient to maintain:
  - Adding e to G:
    $\implies$ few $O(1/\epsilon^4)$ changes to H

Our Techniques

($\mu(G) = |\text{maximum matching}|$)

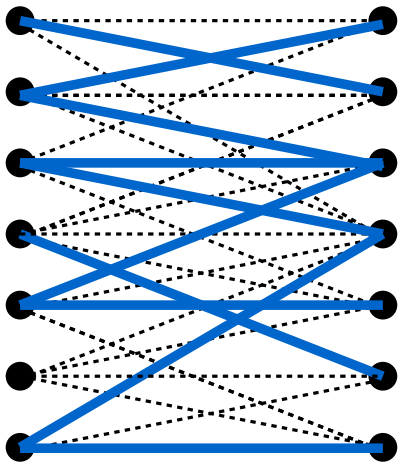


Graph G has many edges...

Keep track of a subgraph H:

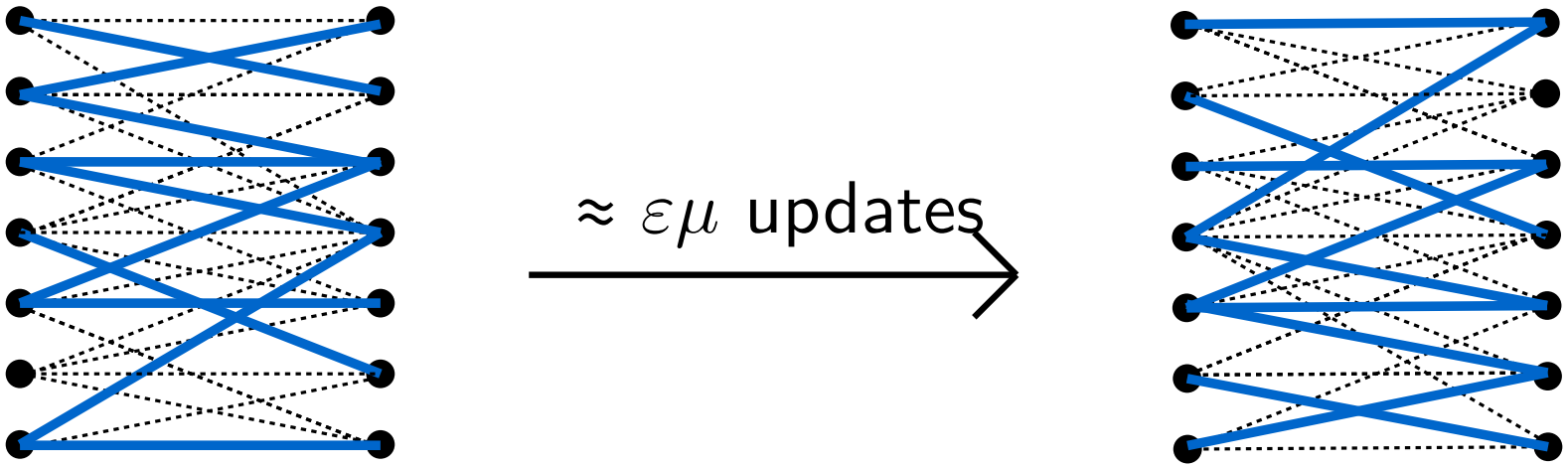
- "EDCS" [Bernstein-Stein ICALP'15]
- Sparse: $|E(H)| \leq \mu(G)/\varepsilon^2$
- Preserves good matching: $\mu(H) \geq (1-\varepsilon)\mu(G)$
- Efficient to maintain:
  - Adding e to G:
    $\implies$ few $O(1/\epsilon^4)$ changes to H
    insertions & deletions

Using the Matching Sparsifier



G, H, and good matching

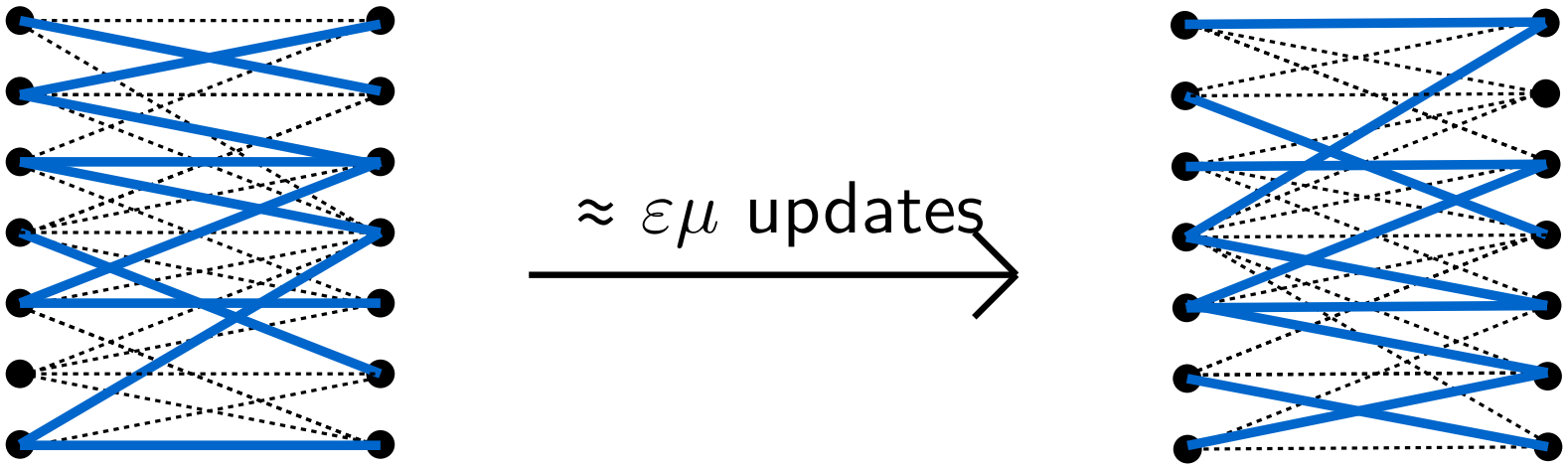
Using the Matching Sparsifier



G, H, and good matching

Be Lazy! [Gupta-Peng FOCS'23]

Using the Matching Sparsifier

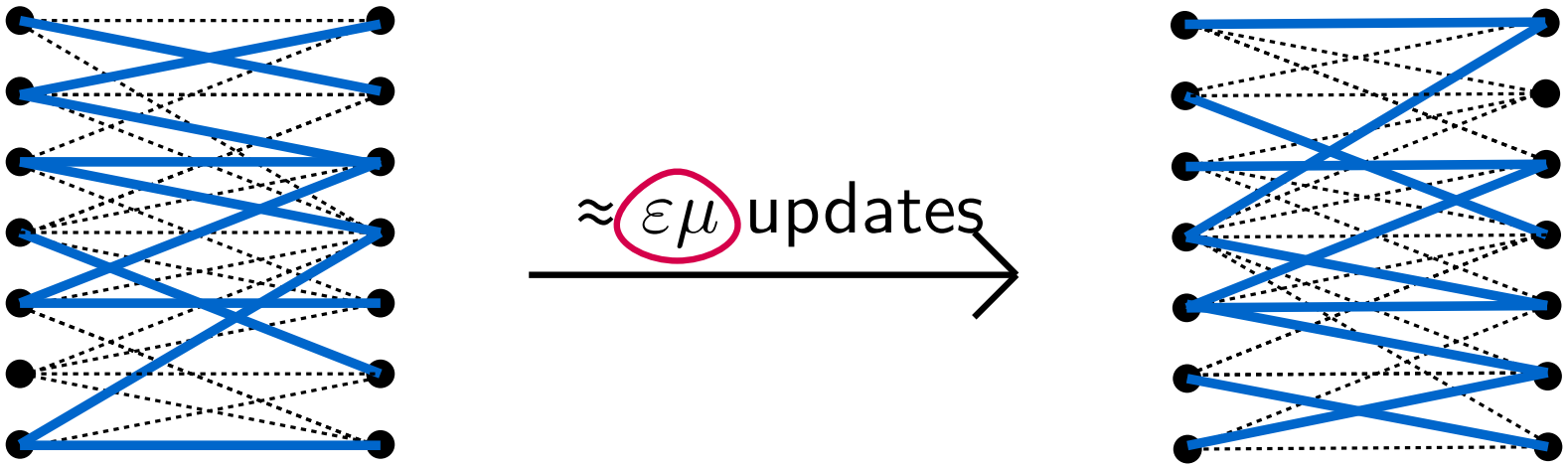


G, H, and good matching

Be Lazy![Gupta-Peng FOCS'23]

Recompute good matching in time $O(|E(H)|/\varepsilon) = O(\mu/\text{poly}(\varepsilon))$

Using the Matching Sparsifier

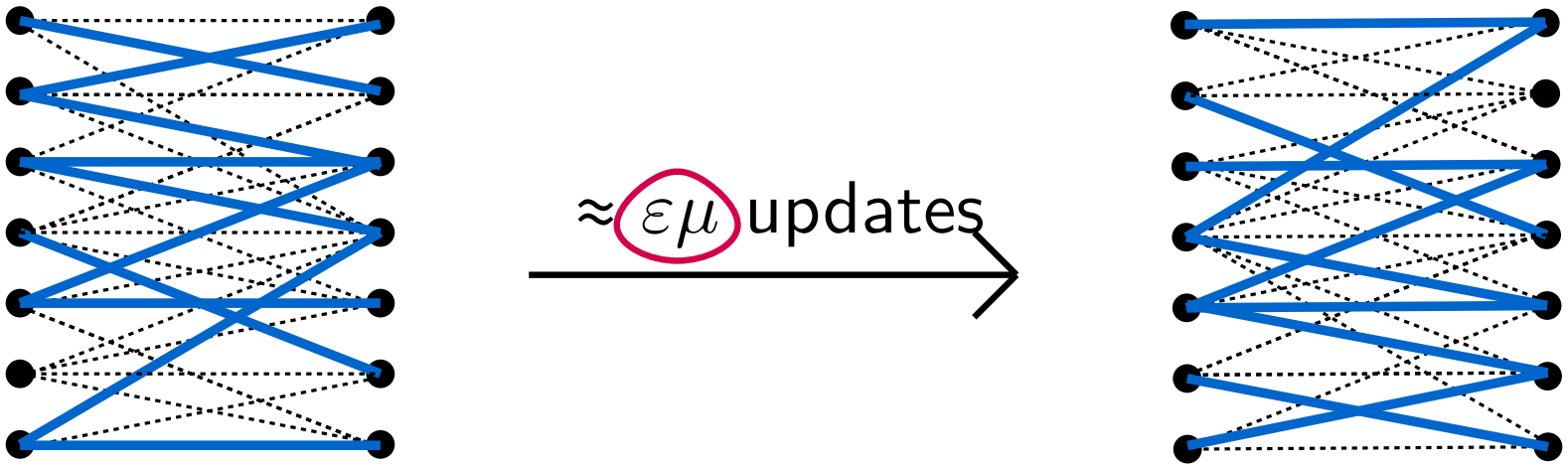


G, H, and good matching

Be Lazy! [Gupta-Peng FOCS'23]

Recompute good matching in time $O(|E(H)|/\varepsilon) = O(\mu/\text{poly}(\varepsilon))$

Using the Matching Sparsifier



G, H, and good matching

Be Lazy! [Gupta-Peng FOCS'23]

Recompute good matching in time $O(|E(H)|/\varepsilon) = O(\mu/\text{poly}(\varepsilon))$

= poly$(1/\varepsilon)$ (amortized) time per update

Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

- $\deg_H(u) + \deg_H(v) \geq \beta - 1$, for all $(u, v) \in E(G)$
- $\deg_H(u) + \deg_H(v) \leq \beta$, for all $(u, v) \in E(H)$
- Edges might appear multiple times in H.

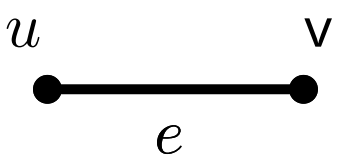
Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

- $\deg_H(u) + \deg_H(v) \geq \beta - 1$, for all $(u, v) \in E(G)$
- $\deg_H(u) + \deg_H(v) \leq \beta$, for all $(u, v) \in E(H)$
- Edges might appear multiple times in H.



Idea: If edge e unhappy $\implies$ fix it!

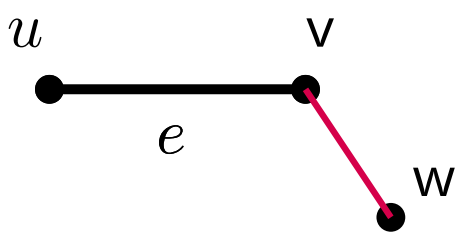
Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

- $\deg_H(u) + \deg_H(v) \geq \beta - 1$, for all $(u, v) \in E(G)$
- $\deg_H(u) + \deg_H(v) \leq \beta$, for all $(u, v) \in E(H)$
- Edges might appear multiple times in H.



Idea: If edge e unhappy $\Longrightarrow$ fix it!

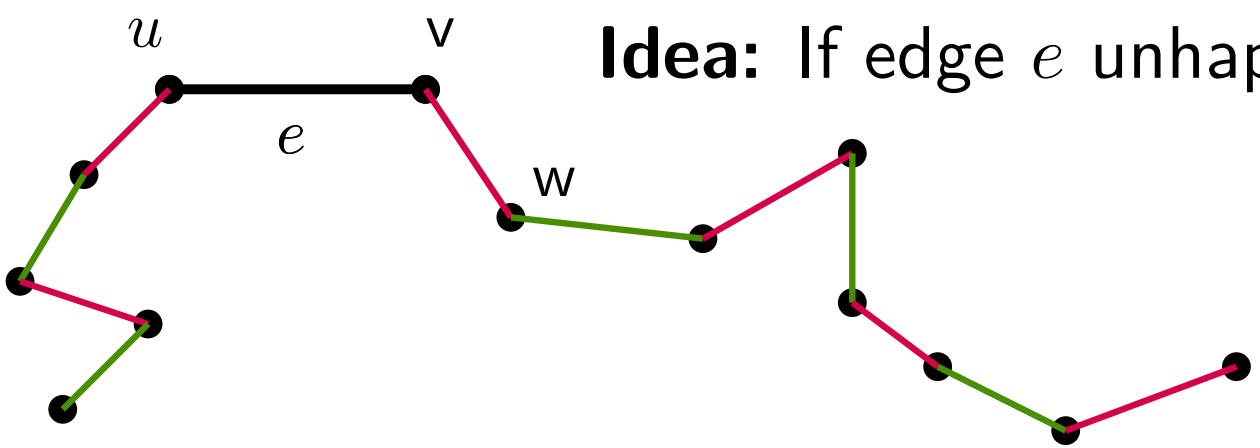
Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

- $\deg_H(u) + \deg_H(v) \geq \beta - 1$, for all $(u, v) \in E(G)$
- $\deg_H(u) + \deg_H(v) \leq \beta$, for all $(u, v) \in E(H)$
- Edges might appear multiple times in H.



Idea: If edge e unhappy $\Longrightarrow$ fix it!

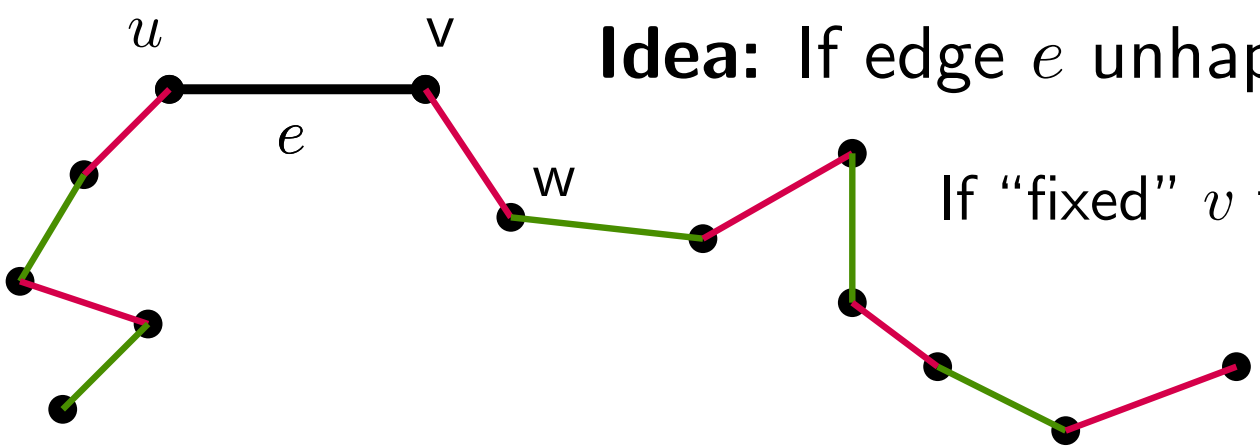
Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H of G: ($\beta = \Theta(\frac{1}{\varepsilon^2})$)

- $\deg_H(u) + \deg_H(v) \geq \beta - 1$, for all $(u, v) \in E(G)$
- $\deg_H(u) + \deg_H(v) \leq \beta$, for all $(u, v) \in E(H)$
- Edges might appear multiple times in H.



Idea: If edge e unhappy $\Longrightarrow$ fix it!

If "fixed" v too many times, ignore it

Weighted Edge-Degree Constrained Subgraph

[Bernstein-Stein ICALP'15]

Dynamic, Streaming, Sublinear, Communication, Fault Tolerant, ...

β-**WEDCS** H

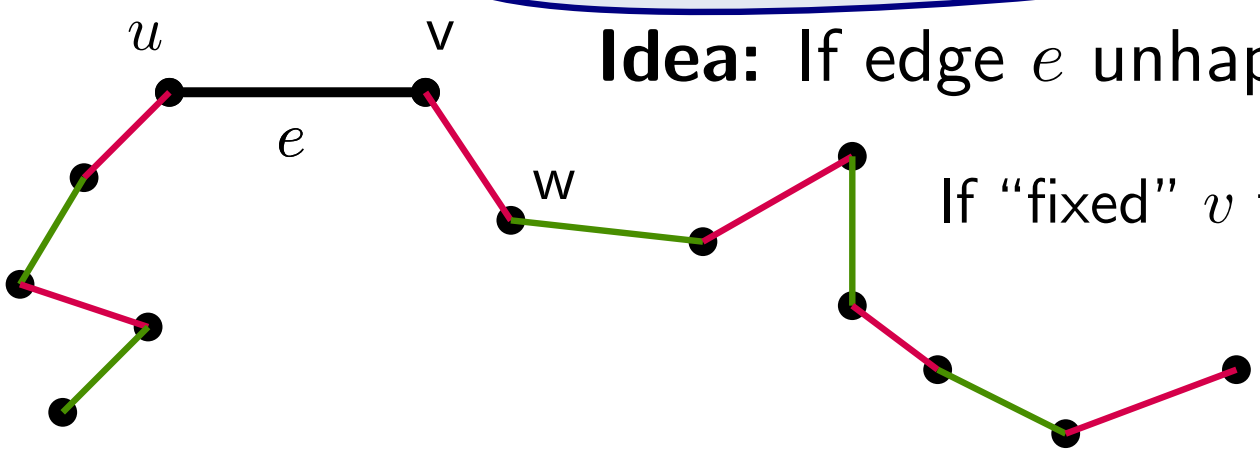
- $\deg_H(u)$ –
- $\deg_H(u)$ –
- Edges migh

Keep track of a subgraph H:

- Sparse: $|E(H)| \leq \mu(G)/\varepsilon^2$
- Preserves good matching: $\mu(H) \geq (1-\varepsilon)\mu(G)$
- Efficient to maintain

$E(G)$

$H)$



Idea: If edge e unhappy $\implies$ fix it!

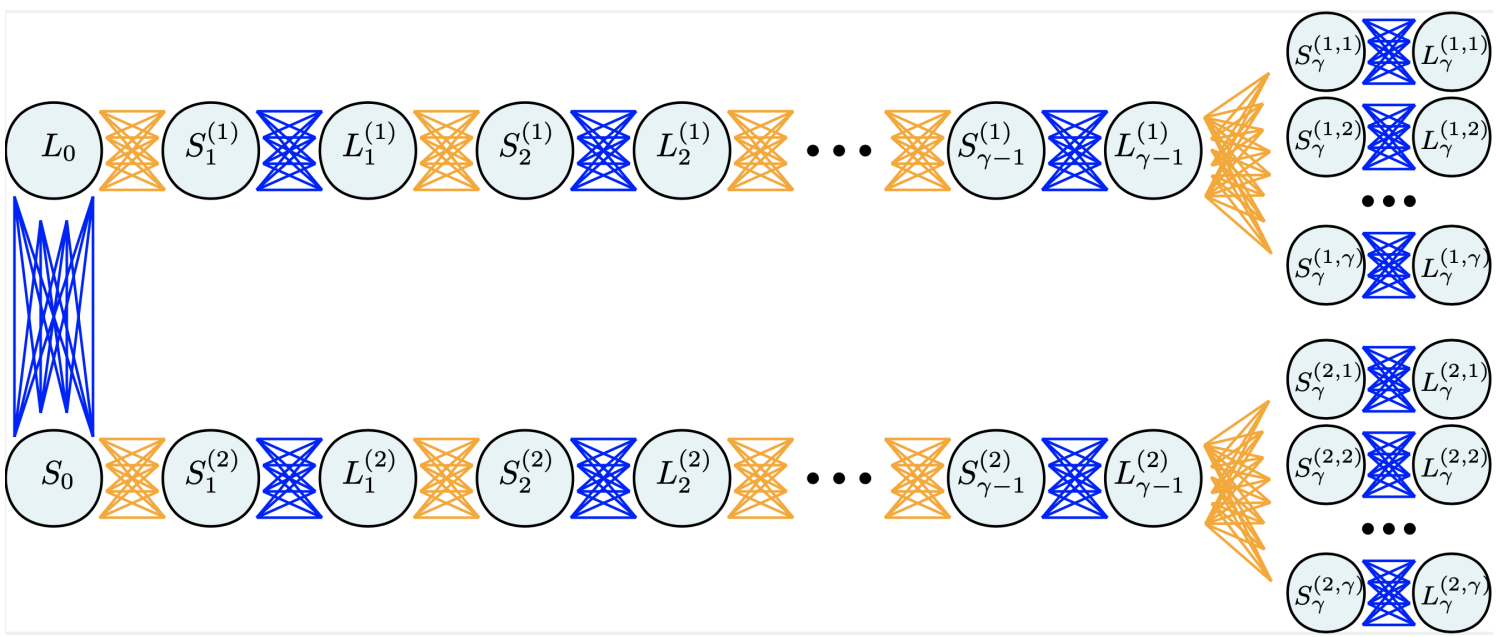
If "fixed" v too many times, ignore it

WEDCS Bonus Results

- Simple approximate *fractional matching* & *vertex cover* on H

$$f_{(u,v)} = \min\left(\frac{1}{\deg_H(u)}, \frac{1}{\deg_H(v)}\right)$$

- Setting $\beta = \Theta(1/\varepsilon^2)$ is tight ($=$ maxdegree in H)



Summary & Open Problems

Summary:

- Incremental (1-ε)-approx bipartite matching in $O(\text{poly}(\frac{1}{\varepsilon}))$ update time
- Using *Weighted Edge-Degree-Constrained-Subgraph* matching sparsifier

Summary & Open Problems

Summary:

- Incremental (1-ε)-approx bipartite matching in $O(\text{poly}(\frac{1}{\varepsilon}))$ update time
- Using *Weighted Edge-Degree-Constrained-Subgraph* matching sparsifier

Open Problems:

- Constant time $(1-\varepsilon)$-approximate decremental matching?
- Constant time $(1-\varepsilon)$-approximate weighted matching?
- Worst-Case instead of Amortized update time?
  - Open for both $(1-\varepsilon)$-approx incremental & decremental.
  - Also when allowing polylog(n) dependence.
- More applications of (W)EDCS in other models of computation?

Summary & Open Problems

Summary:

- Incremental (1-ε)-approx bipartite matching in $O(\text{poly}(\frac{1}{\varepsilon}))$ update time
- Using *Weighted Edge-Degree-Constrained-Subgraph* matching sparsifier

Open Problems:

- Constant time $(1-\varepsilon)$-approximate decremental matching?
- Constant time $(1-\varepsilon)$-approximate weighted matching?
- Worst-Case instead of Amortized update time?
  - Open for both $(1-\varepsilon)$-approx incremental & decremental.
  - Also when allowing polylog(n) dependence.
- More applications of (W)EDCS in other models of computation?

Thanks!