Toward Optimal Semi-streaming Algorithm for (1+ε)-approximate Maximum Matching

Slobodan Mitrović
(UC Davis)

Anish Mukherjee
(University of Warwick)

Piotr Sankowski
(University of Warsaw)

Wen-Horng Sheu
(UC Davis)

Maximum matching problem

- Let $G = (V, E)$ be an unweighted graph
- A *matching* is a set of edges that do not share an endpoint
- **Goal:** Find the largest matching

Prior work

- The problem is extensively studied in different settings:
- **Polynomial time:** [Berge ‘57] [Edmonds ‘65] [Hopcroft, Karp ‘73] [Micali, Vazirani ‘80] [Gabow ‘90] [Kalantari, Shokoufandeh ‘95] …
- **Estimating size in streaming:** [Kapralov, Khanna, Sudan ‘14] [Assadi, Khanna, Li ‘17] [Kapralov, Mitrović, Norouzi-Fard, Tardos ‘20] …
- **Dynamic:** [Bernstein, Stein ‘16] [Solomon ‘16] [Bhattacharya, Kulkarni ‘19] [Behnezhad, Łącki, Mirrokni ‘19] [Behnezhad, Khanna ‘22] …
- **Semi-streaming:** [McGregor ‘05] [Ahn, Guha, '11] [Eggert, Kliemann, Munstermann, Srivastav, '12] [Ahn, Guha, '13] [Kapralov, '13] [Ahn, Guha, '18] [Tirodkar, '18] [Gamlath, Kale, Mitrović, Svensson, '19] [Assadi, Liu, Tarjan, '21] [Assadi, Jambulapati, Jin, Sidford, Tian, '22] [Fischer, Mitrović, Uitto, '22] [Huang, Su, '23] [Assadi, '24]

Semi-streaming setting

- No random access to the input graph
- Edges are presented as a stream, arriving in arbitrary order
- Reading the stream once is called a pass

- The algorithm can use ***O*(*n* poly log *n*)** memory.
- Allowed to make multiple passes over the stream.

- **Goal:** minimize the number of passes
- **Problem:** finding a $(1+\varepsilon)$-approximate maximum matching
  (on general graphs)

Prior work

- "Constant number" of passes is achievable
- [McGregor '05]: $(1/\varepsilon)^{O(1/\varepsilon)}$ passes

- Dependence on ε has been improved since then

- Two classes of graphs:
  - bipartite
  - general

- Two families of studies:
  - **constant-pass:** complexity only depends on **1/ε** (our focus)
  - **ε-efficient:** complexity depends on **log *n*** and **1/ε**

Prior work (bipartite)

- poly(1/ε) is known since 2009 [Eggert, Kliemann, Munstermann, Srivastav]

Source	Pass	Weighted?
[McG, 2005]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[EKS, 2009]	$1/\varepsilon^8$	
[EKMS, 2012]	$1/\varepsilon^5$	
[AG, 2013]	$1/\varepsilon^5 \cdot \log(1/\varepsilon)$	Yes
[Kap, 2013]	$1/\varepsilon^2$ (vertex arrival)	
[AG, 2018]	$\log(n) / \varepsilon$	Yes
[ALT, 2021]	$\mathbf{1 / \varepsilon^2}$	
[AJJST, 2022]	$\log(n) / \varepsilon \cdot \log(1/\varepsilon)$	
[Ass, 2024]	$\log(n) / \varepsilon$	Yes

Prior work (general)

- poly($1/\varepsilon$) is only known recently [Fischer, Mitrović, Uitto, 2022]
- Huge gap between bipartite and general graphs
- Bipartite graphs: $1/\varepsilon^2$ passes [ALT21]
- General graphs: $1/\varepsilon^{19}$ passes [FMU22]

Source	Pass	Weight?
[McG, 2005]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[AG, 2011]	$\log(n) / \varepsilon^7 \cdot \log(1/\varepsilon)$	
[AG, 2013]	$\log(n) / \varepsilon^4$	Yes
[AG, 2018]	$\log(n) / \varepsilon$	Yes
[Tir, 2018]	$\exp(1/\varepsilon)$	
[GKMS, 2019]	$\exp(1/\varepsilon^2)$	Yes
[FMU, 2022]	$\mathbf{1/\varepsilon^{19}}$	
[HS, 2023]	poly($1/\varepsilon$) but $> 1/\varepsilon^{19}$	Yes
[Ass, 2024]	$\log(n) / \varepsilon$	Yes

Our result

- A $1/\varepsilon^6$-pass algorithm
- Bridging the gap between bipartite and general graphs
- Simpler approach
- Simpler analysis

Source	Pass	Weight?
[McG '05]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[AG11]	$\log(n) / \varepsilon^7 \cdot \log(1/\varepsilon)$	
[AG13]	$\log(n) / \varepsilon^4$	Yes
[AG18]	$\log(n) / \varepsilon$	Yes
[AG18]	$\log(n) / \varepsilon$	Yes
[Tir18]	$\exp(1/\varepsilon)$	
[GKMS19]	$\exp(1/\varepsilon^2)$	Yes
[FMU22]	$1/\varepsilon^{19}$	
[HS23]	more than $1/\varepsilon^{19}$	Yes
[Ass24]	$\log(n) / \varepsilon$	Yes
[this talk]	$\mathbf{1/\varepsilon^6}$	

Remark: other models

- Our algorithm can be simulated in other computational models
- Improve round complexity in MPC and CONGEST models by ε^{-13} factor

Warm-up:
Bipartite graphs

Based on [Eggert, Kliemann, Munstermann, Srivastav ’12]

Definition

- *Free node*: unmatched vertex
- *Alternating path*: path alternates between matched and unmatched edges
- *Augmenting path*: alternating path from a free node to another

Starting point - short augmenting paths

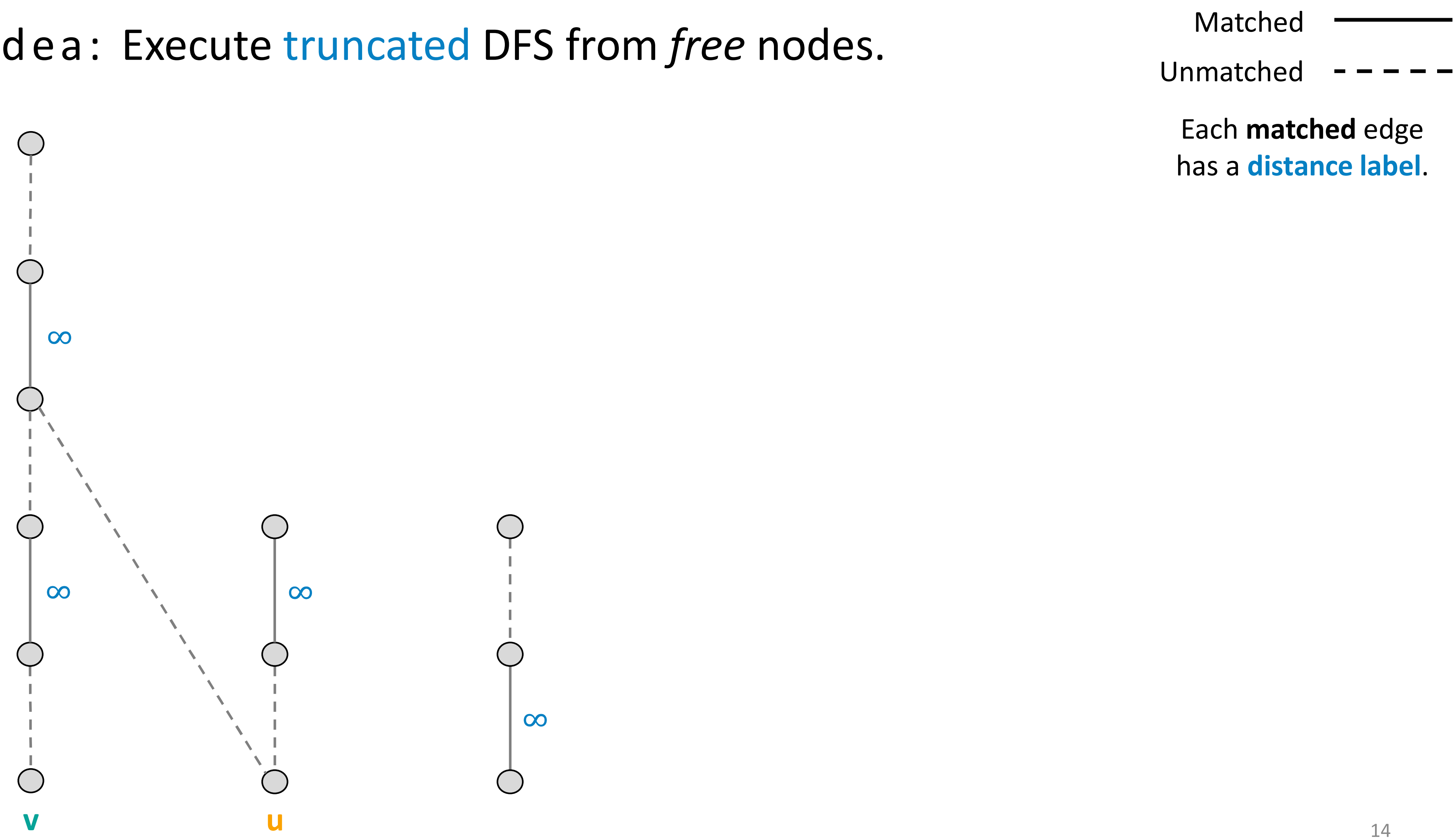
Claim

Let M be a matching and Y be an inclusion-maximal set of $2/\varepsilon$-long augmenting paths. If $|Y| < \varepsilon^2|M|/6$, then M is a $(1+\varepsilon)$-approximate maximum matching.

[Kalantari, Shokoufandeh '95] [McGregor '05] [Eggert, Kliemann, Munstermann, Srivastav '12]

Idea: Execute truncated DFS from *free* nodes.

Idea: Execute truncated DFS from *free* nodes.



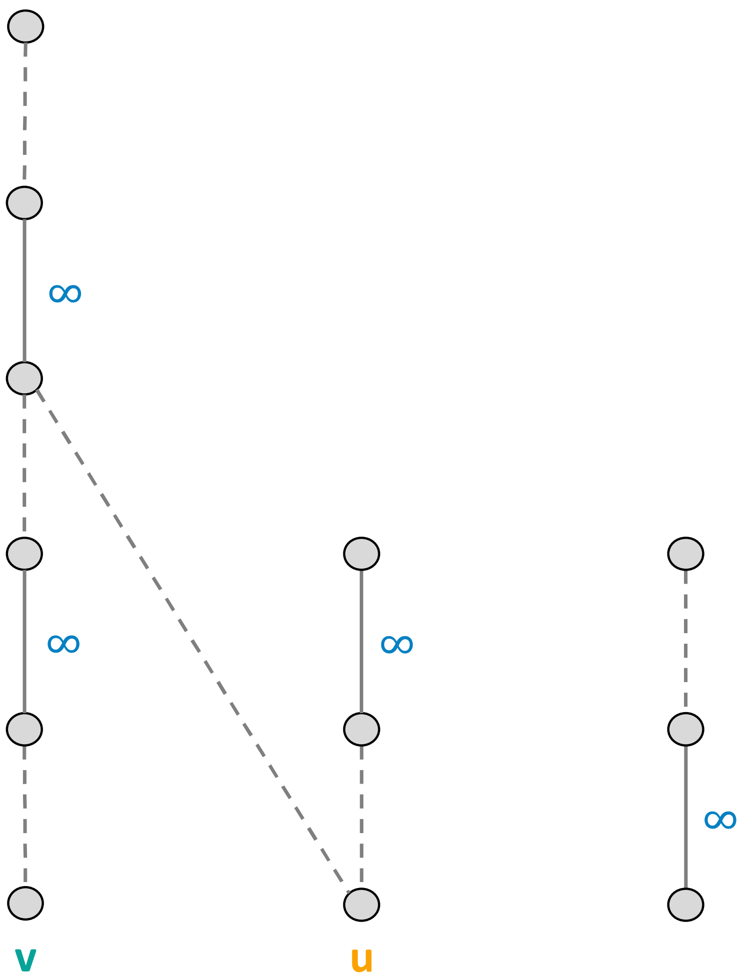
Each **matched** edge has a **distance label**.

Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge has a **distance label**.



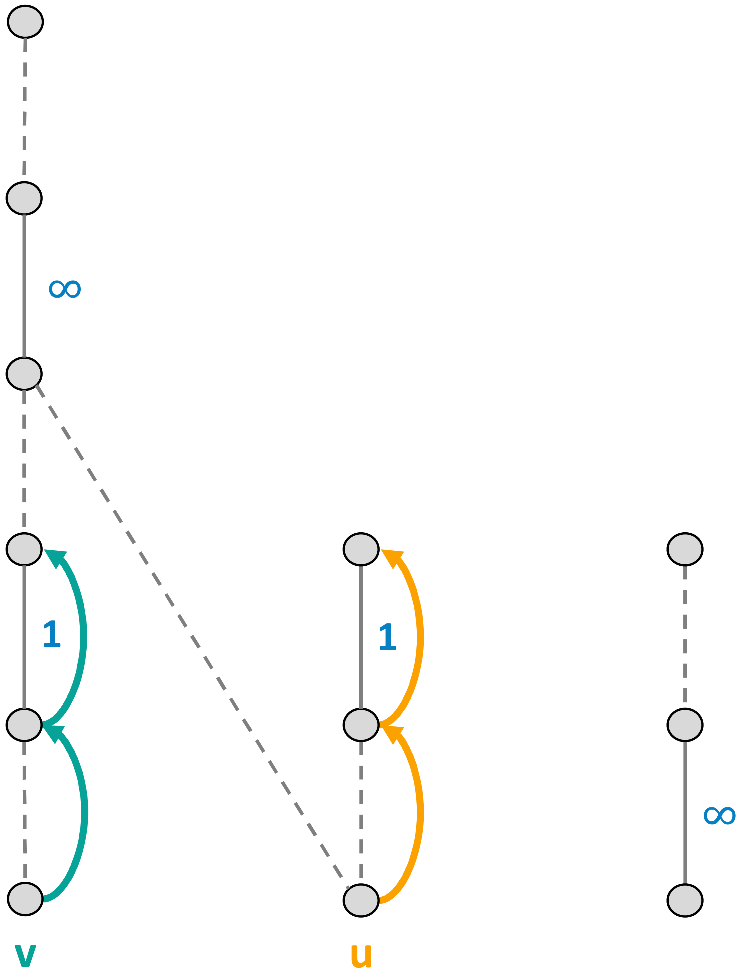
- Meaning of label: current shortest distance
- Each free node maintains an **active path** (DFS search path)

Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge has a **distance label**.



- Meaning of label: current shortest distance
- Each free node maintains an **active path** (DFS search path)

Each pass: Extend by length-2 paths

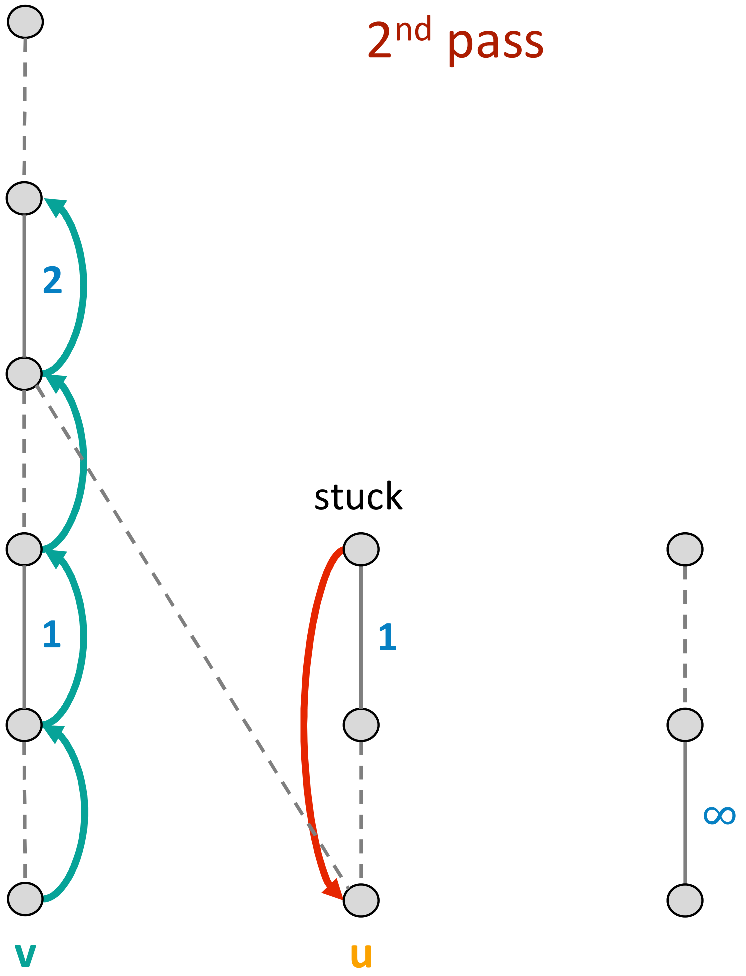
- Scan unmatched edges
- Extend when distance label can be reduced

Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge has a **distance label**.



- Meaning of label: current shortest distance
- Each free node maintains an **active path** (alternating path)

Each pass: Extend by length-2 paths

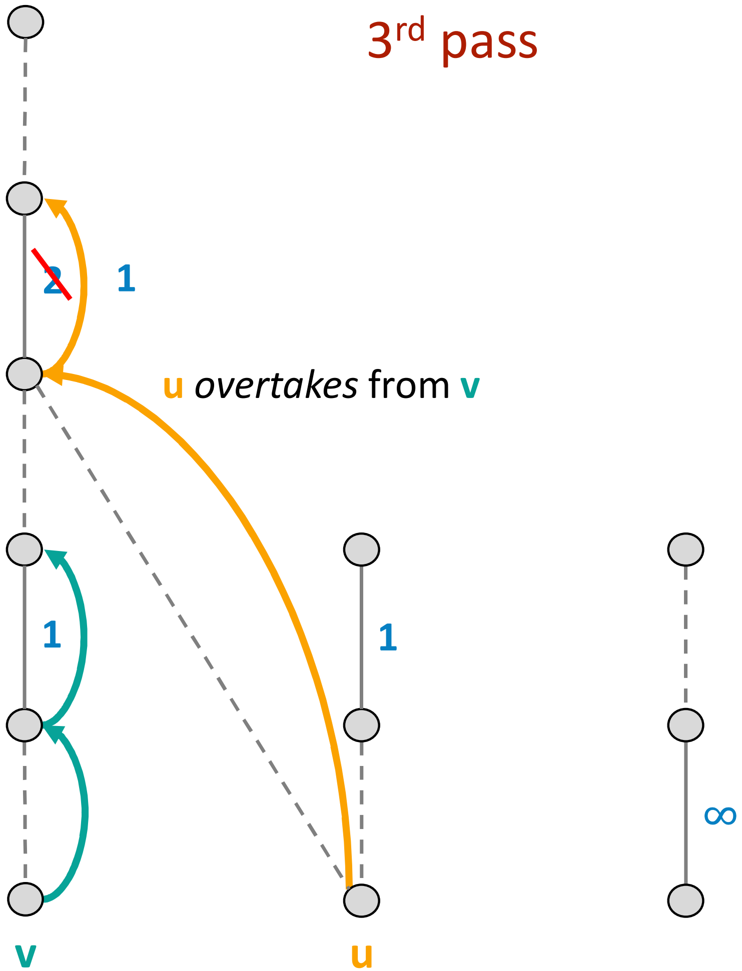
- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck

Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge has a **distance label**.



- Meaning of label: current shortest distance
- Each free node maintains an **active path** (alternating path)

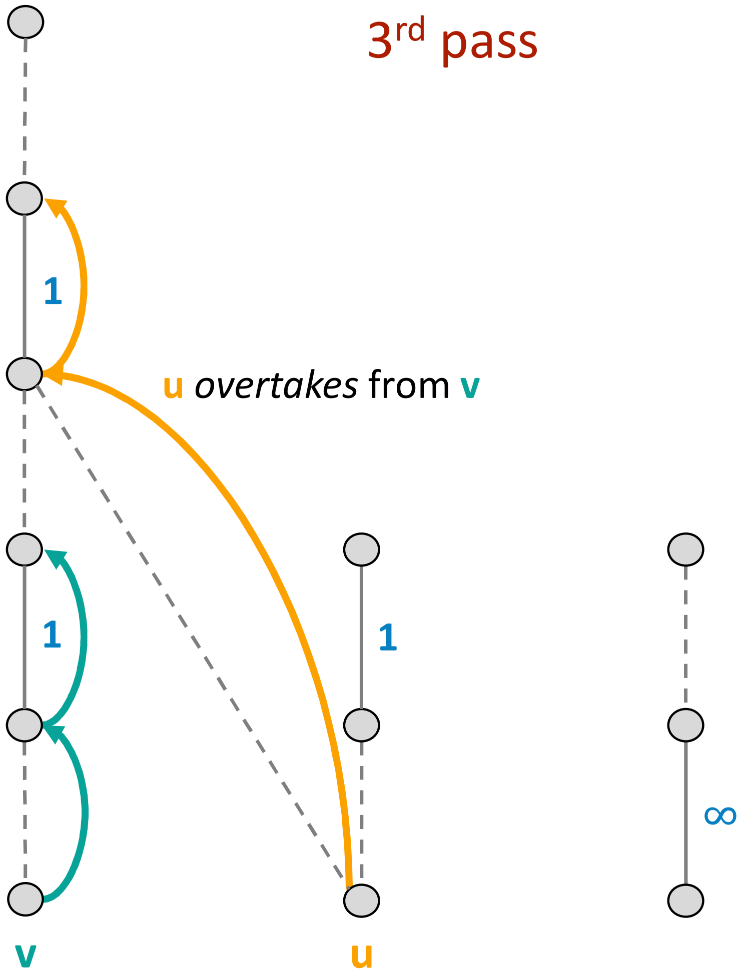
Each pass: Extend by length-2 paths

- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

Idea: Execute truncated DFS from *free* nodes.

Matched
Unmatched

Each **matched** edge has a **distance label**.



- Meaning of label: current shortest distance
- Each free node maintains an **active path** (alternating path)

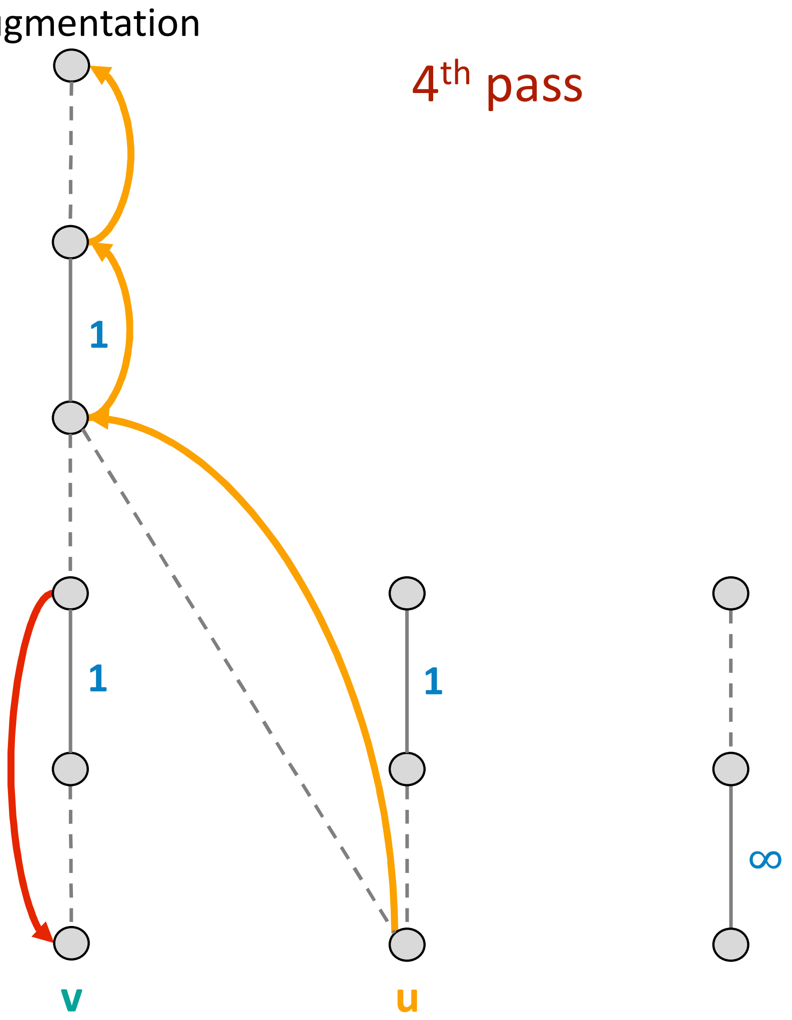
Each pass: Extend by length-2 paths
- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge
has a **distance label**.



- Meaning of label: current shortest distance
- Each free node maintains an **active path**
(alternating path)

Each pass: Extend by length-2 paths

- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

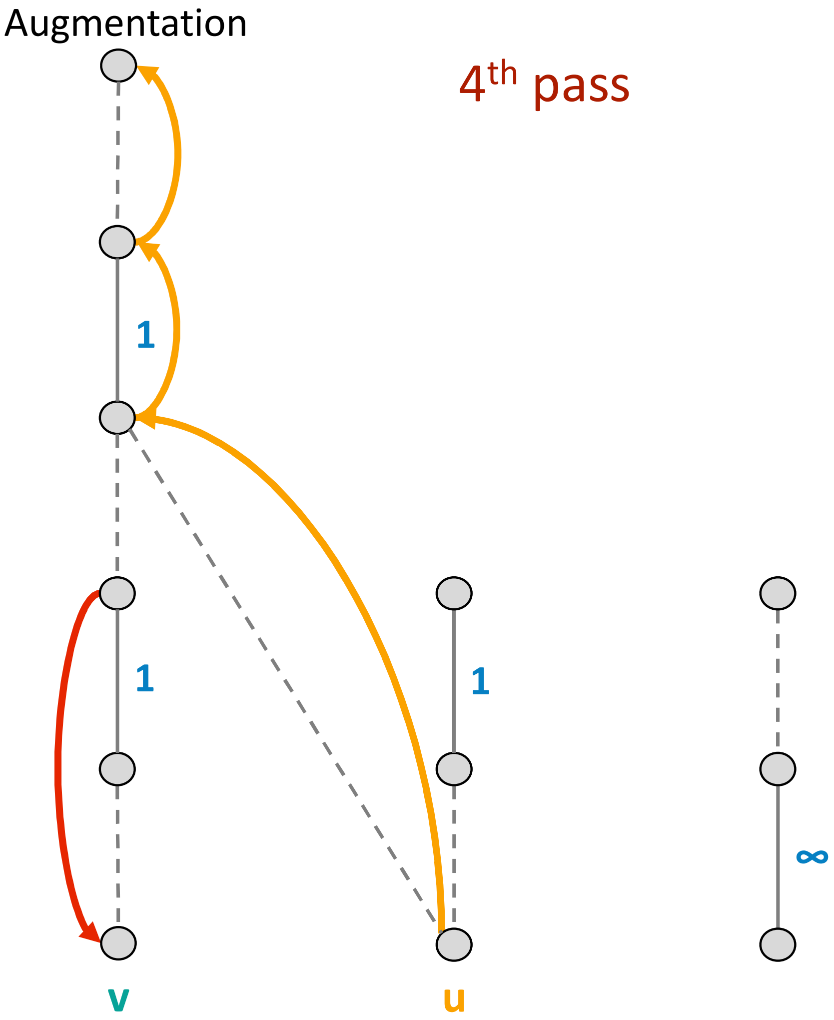
Idea: Execute truncated DFS from *free* nodes.

Matched

Unmatched

Each **matched** edge has a **label**.

4th pass



Analysis

- Run in poly(1/ε) passes
- Find an "almost" maximal set of short augmenting paths

Idea: Execute truncated DFS from *free* nodes.

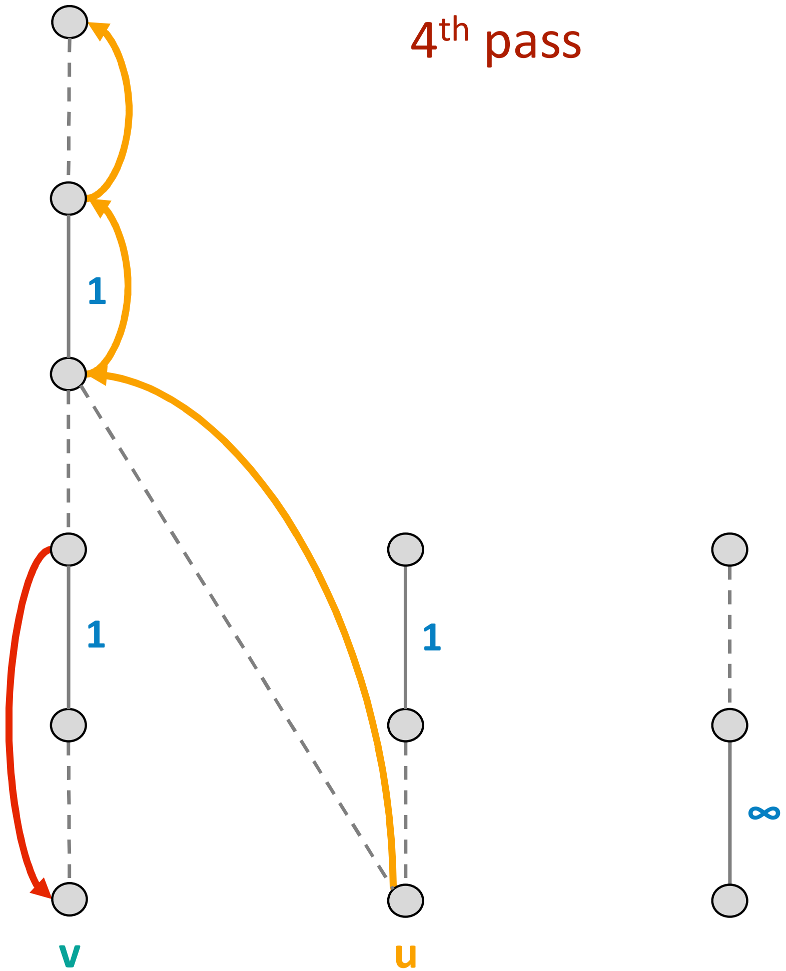
Matched

Unmatched

Each **matched** edge has a **label**.

Augmentation

4^{th} pass



Why poly($1/\varepsilon$) passes?

1. Each matched edge changes label at most $1/\varepsilon$ times
   $\rightarrow$ at most $O(|M| \times 1/\varepsilon)$ label changes and backtrack

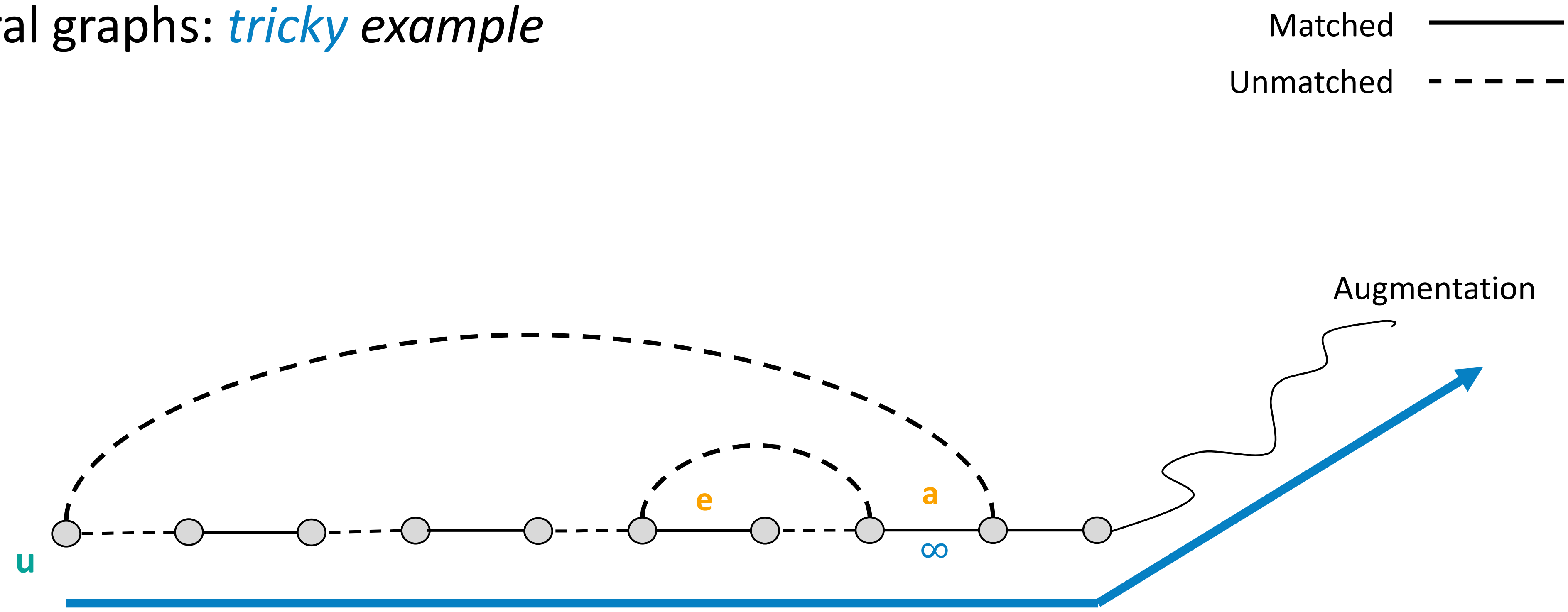
2. Stop when $< \varepsilon^2 |M|$ active free nodes
   $\rightarrow$ at least $\Theta(\varepsilon^2 |M|)$ label changes/backtrack in each pass

3. $O(1/\varepsilon^3)$ passes

General graphs

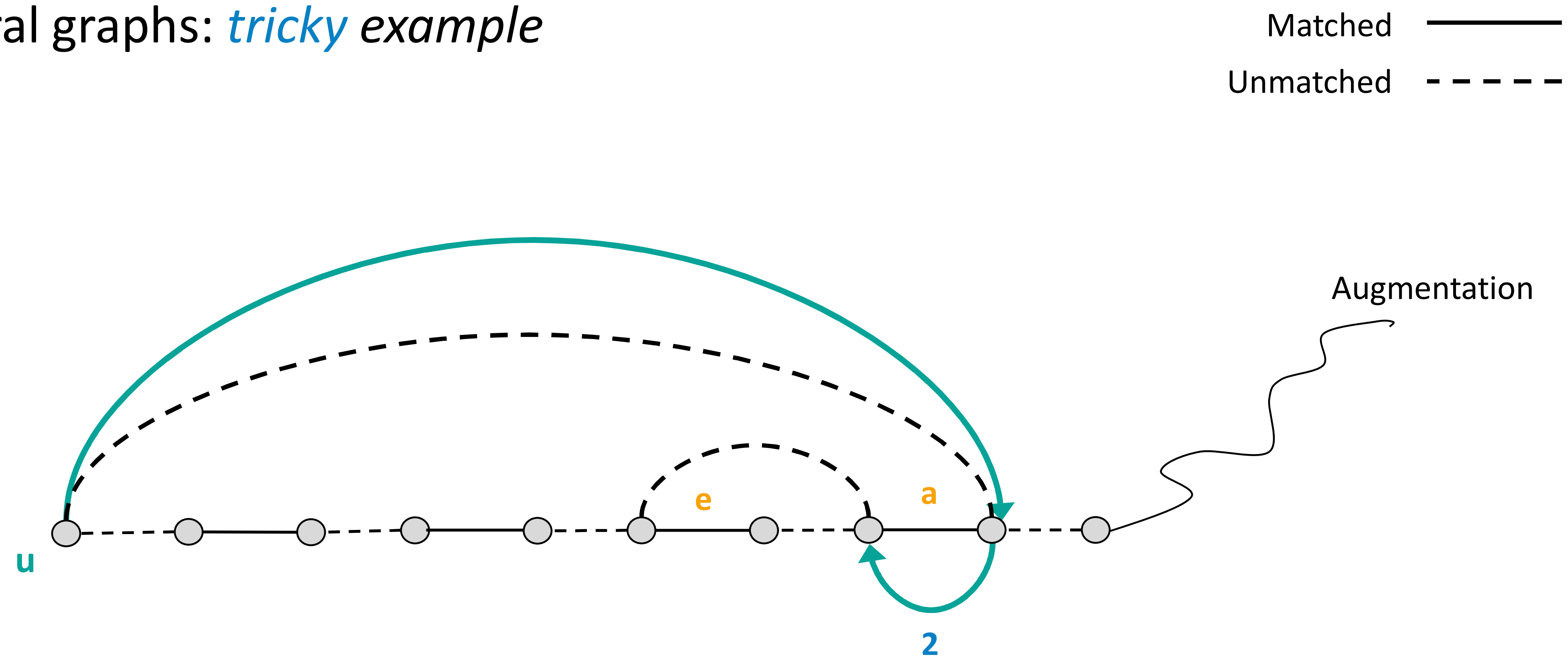
Free node can block itself due to odd cycles

General graphs: *tricky example*

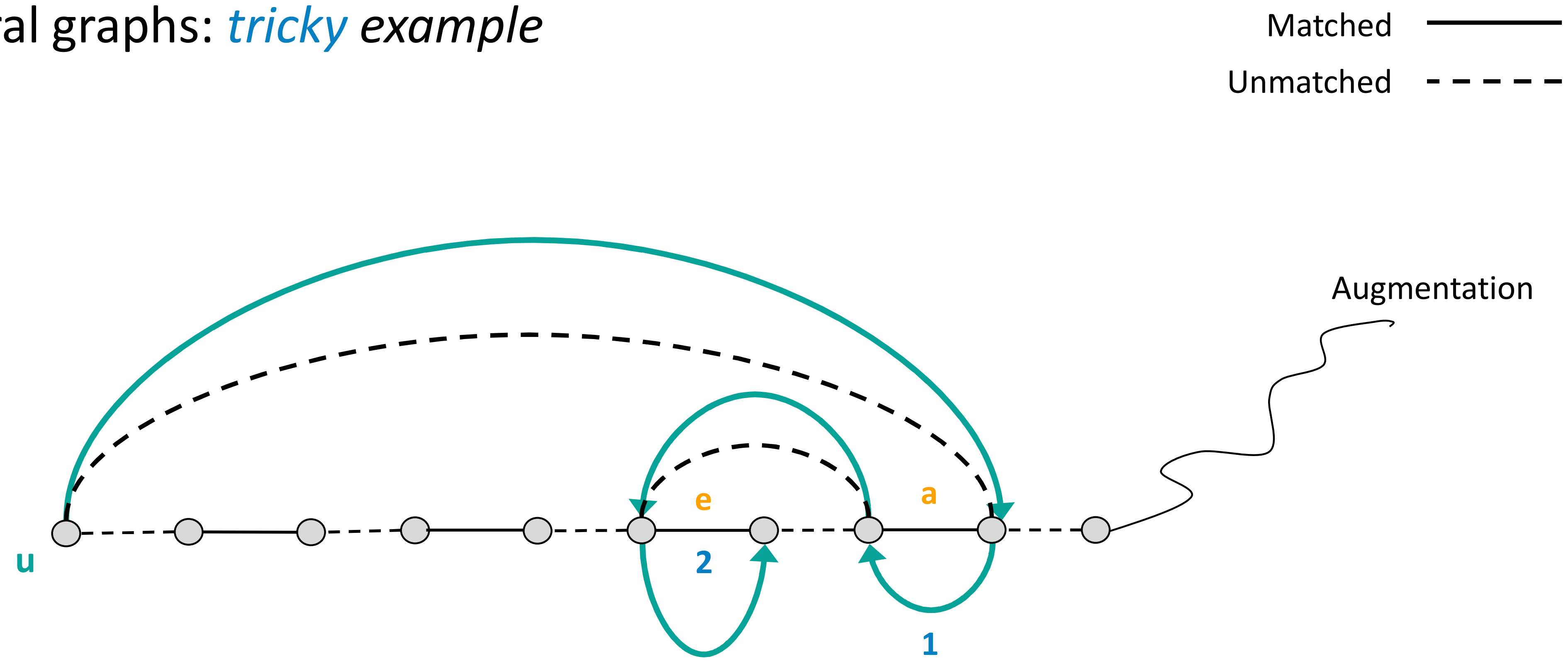


Goal: find this augmentation

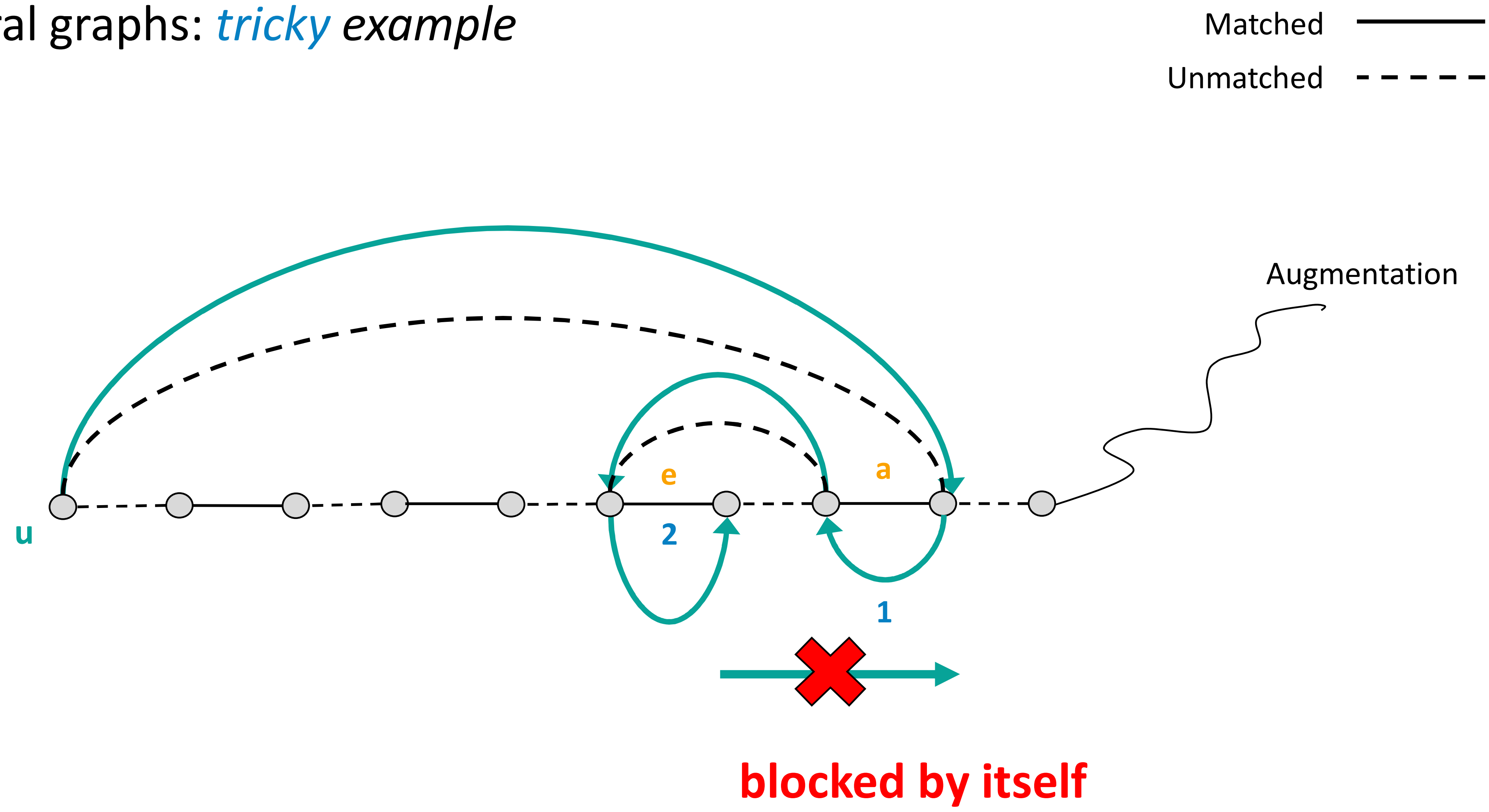
General graphs: *tricky example*



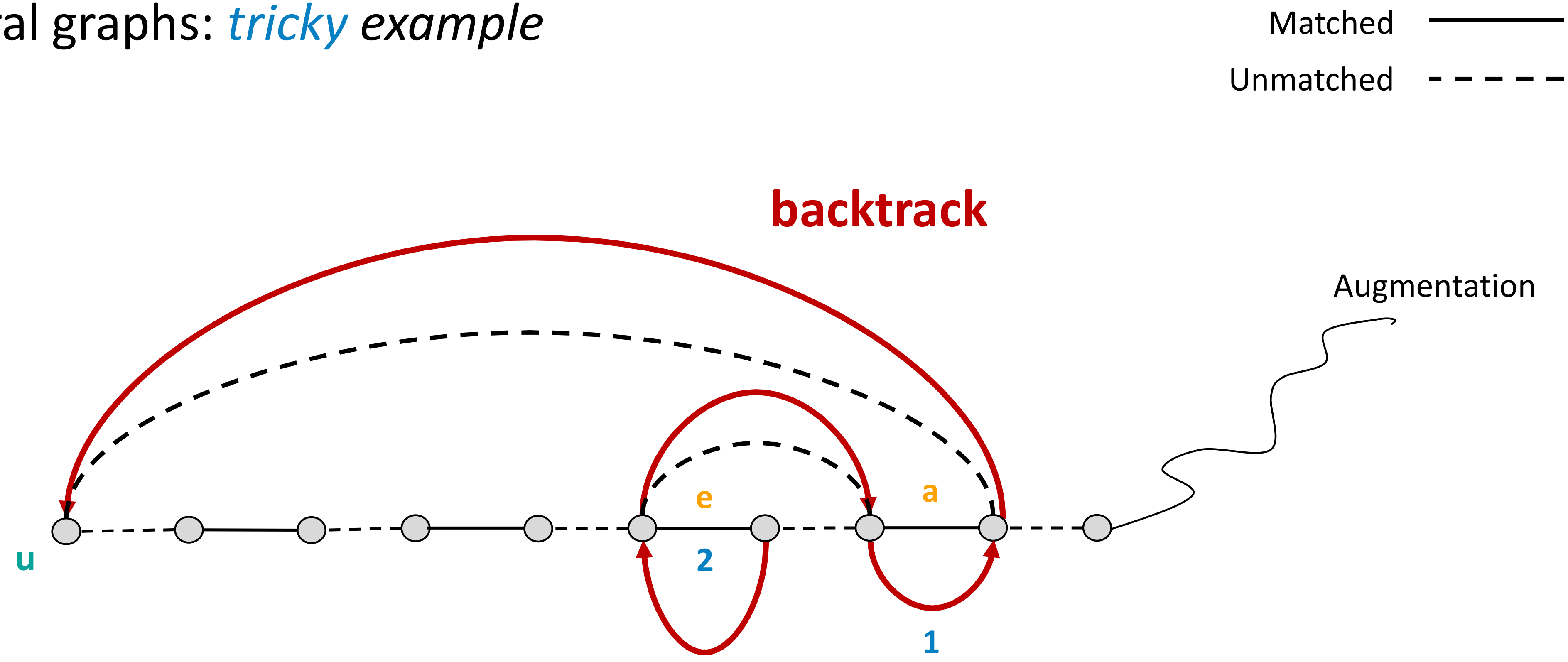
General graphs: *tricky example*



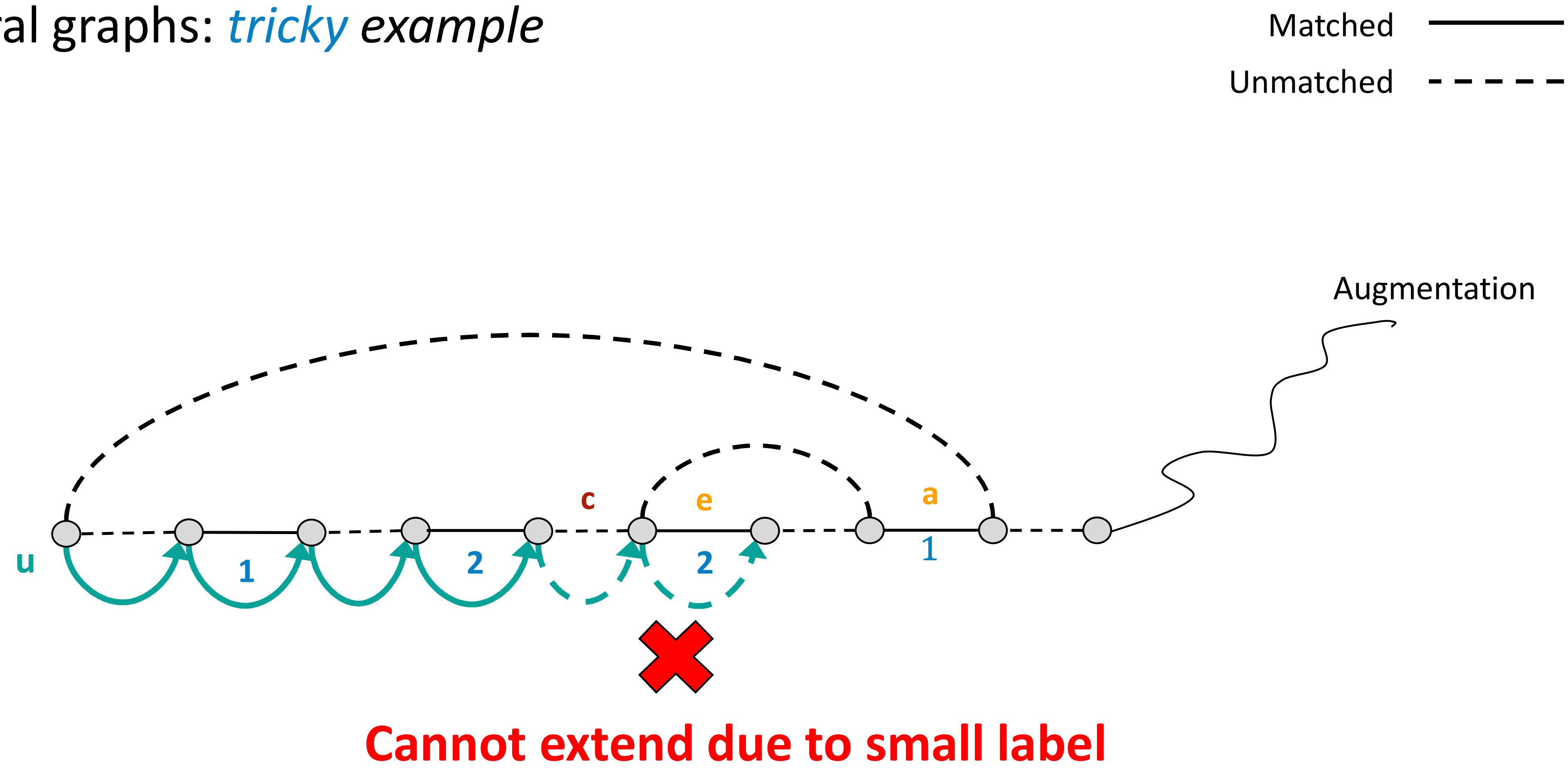
General graphs: *tricky example*



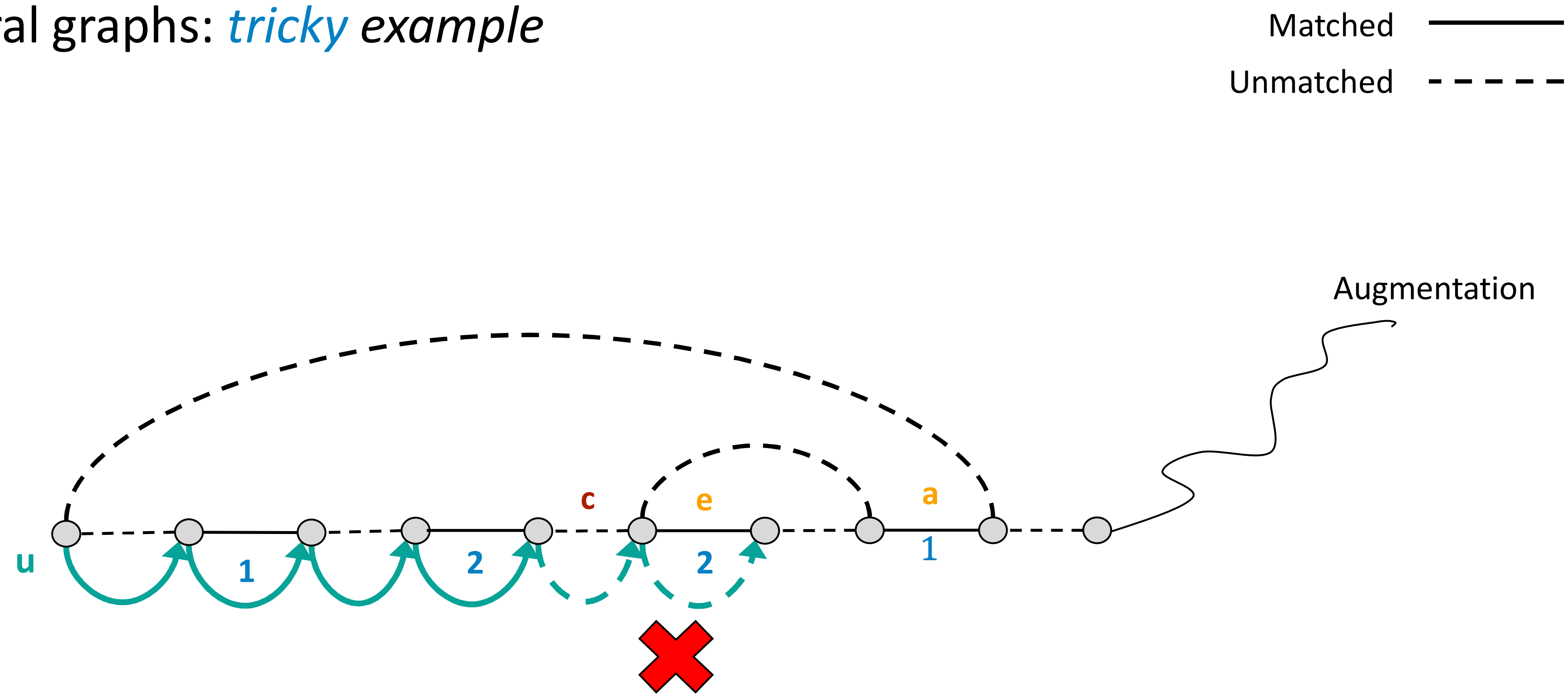
General graphs: *tricky example*



General graphs: *tricky example*



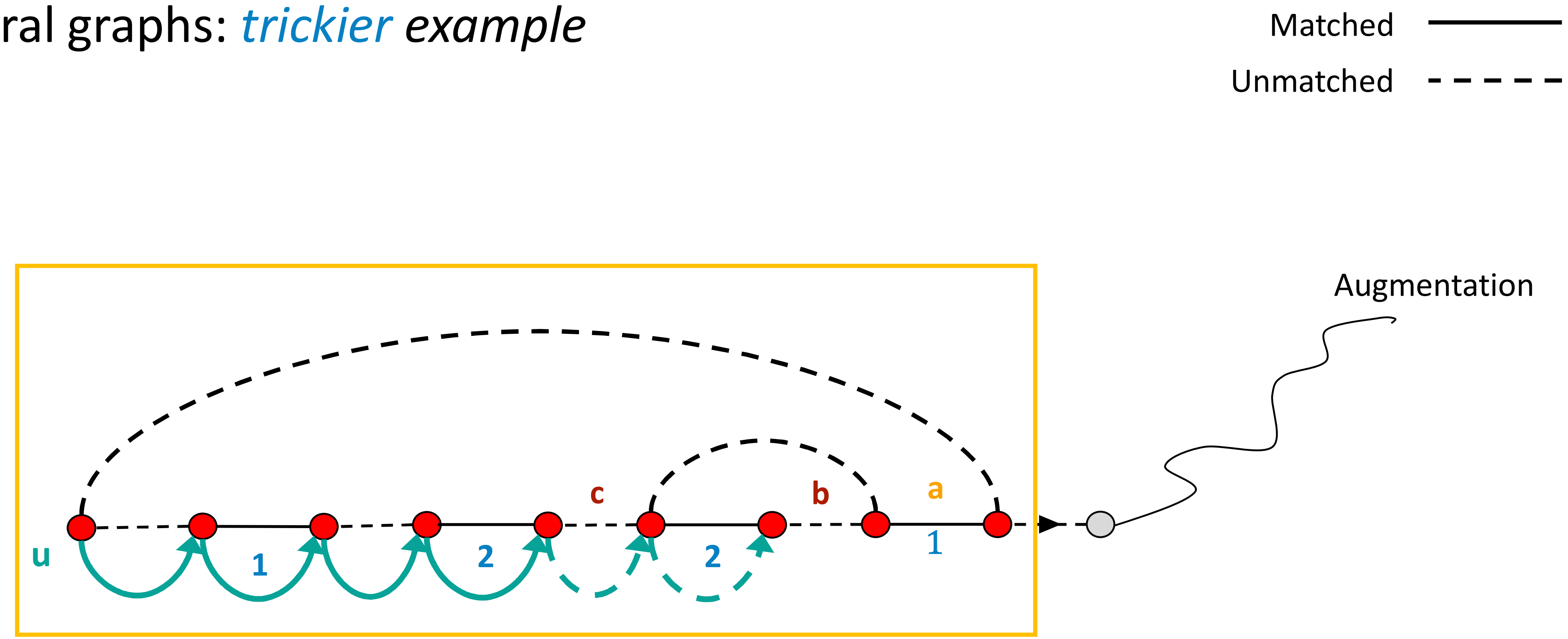
General graphs: *tricky example*



Cannot extend due to small label

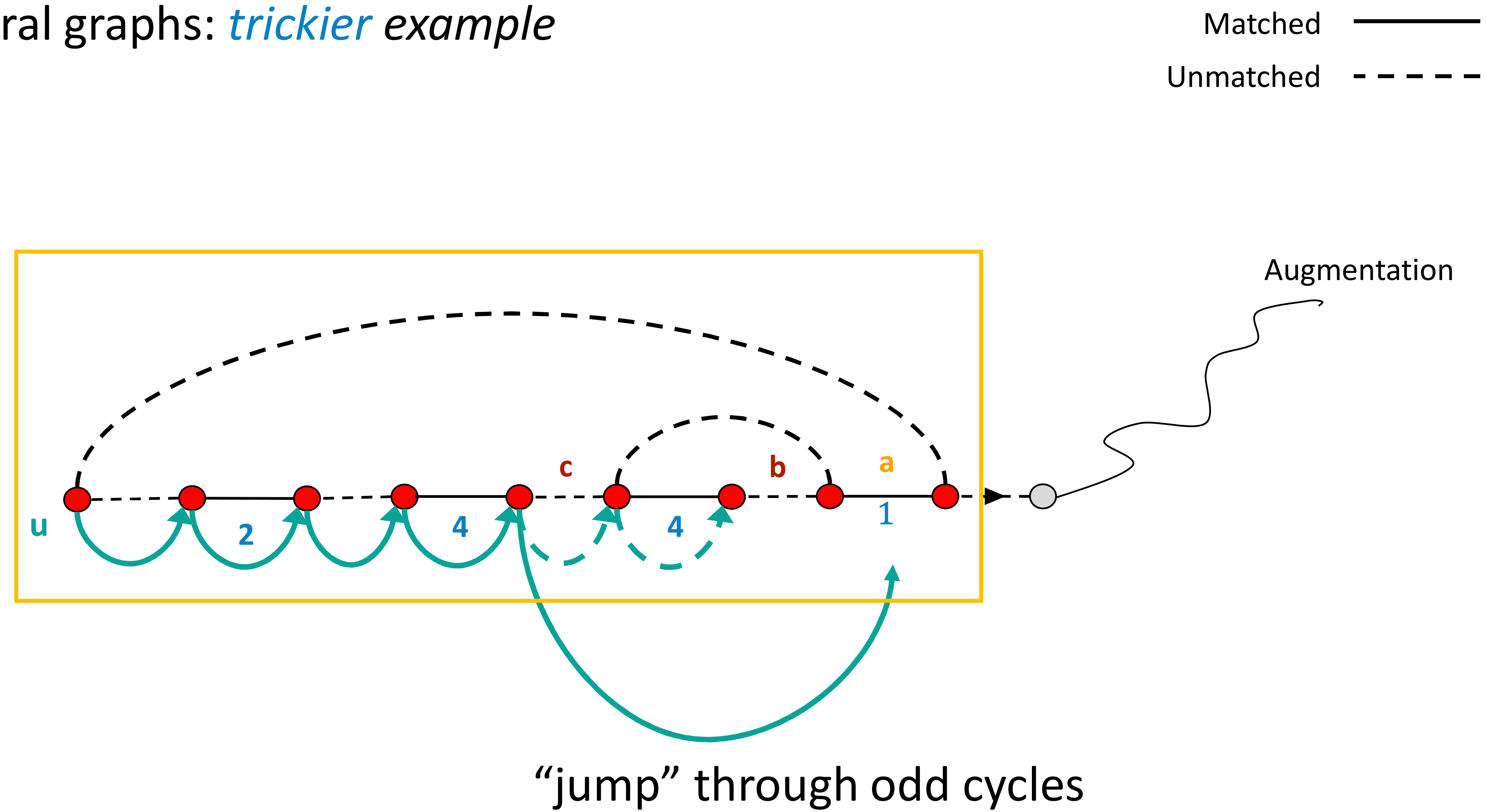
→ augmentation is never found

General graphs: *trickier example*



[FMU]'s approach:
Store all visited vertices and edges
to detect odd cycles

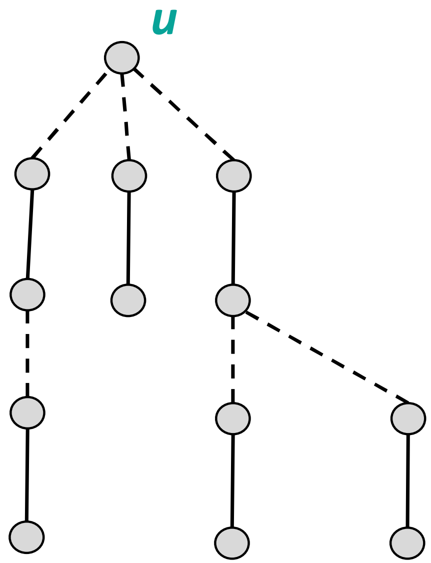
General graphs: *trickier example*



General graphs: our approach

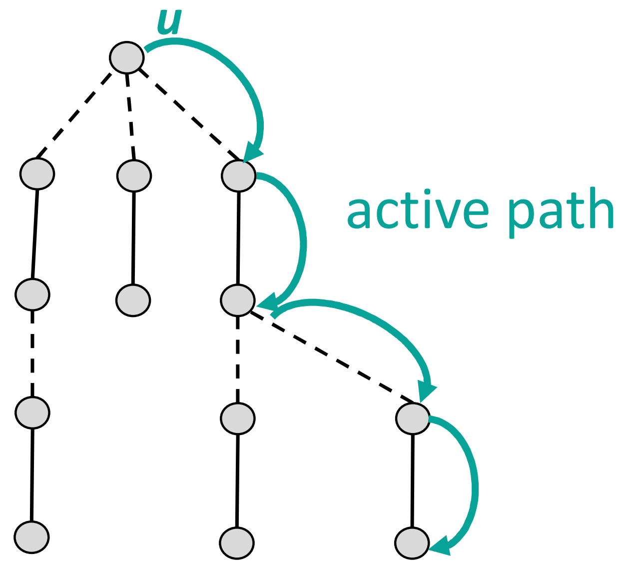
Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint



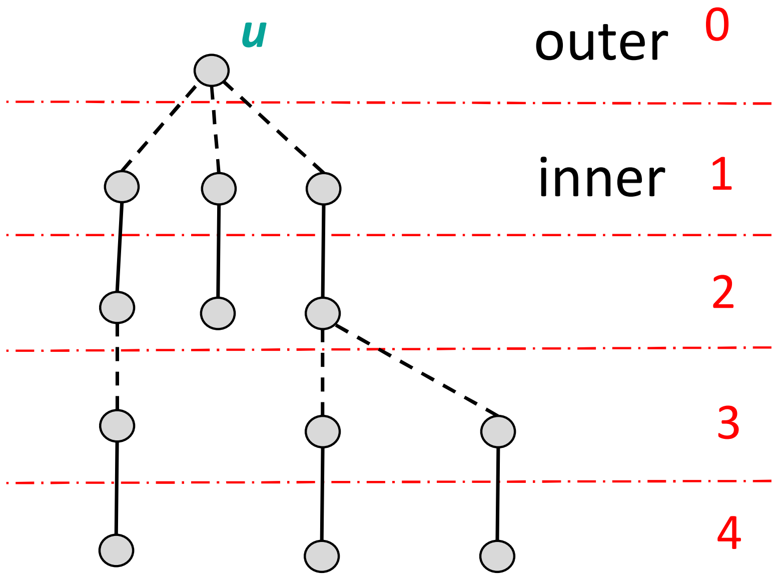
Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint
- Each tree has an active path



Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint
- Each tree has an active path

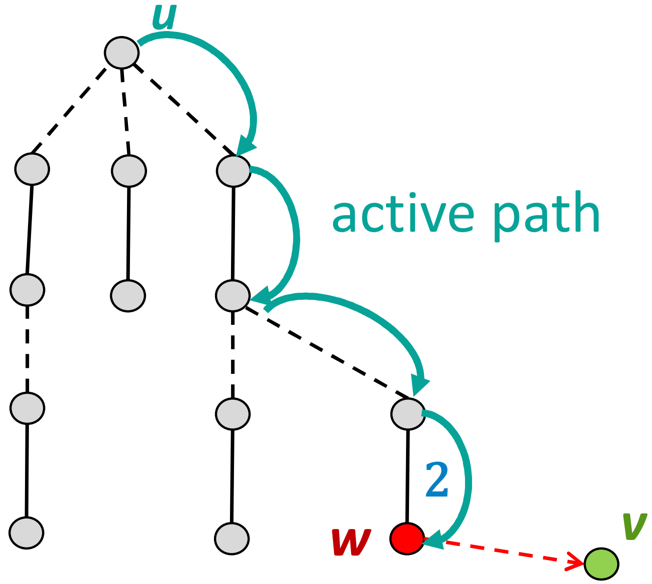


- Even layers: outer vertices
- Odd layers: inner vertices

- Root: outer vertex

Idea: Maintain alternating trees

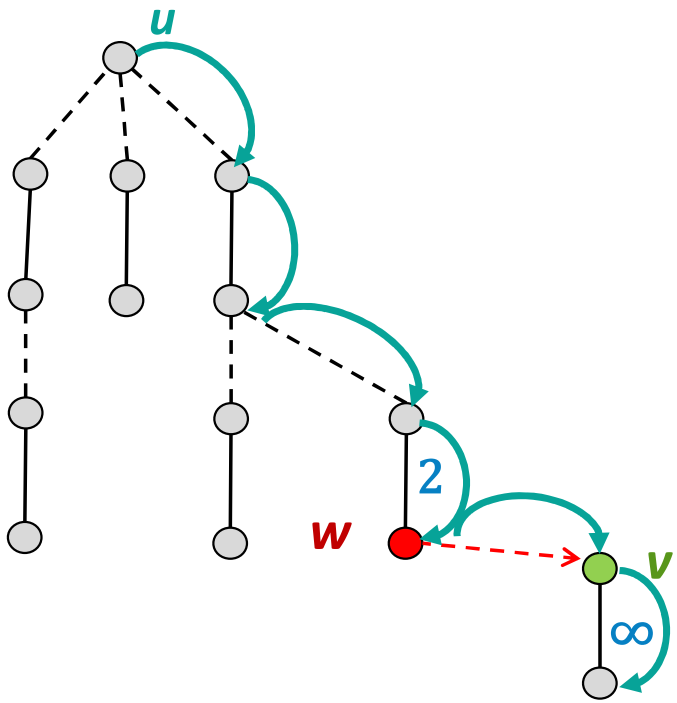
- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path



Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree $\rightarrow$ **Extend**

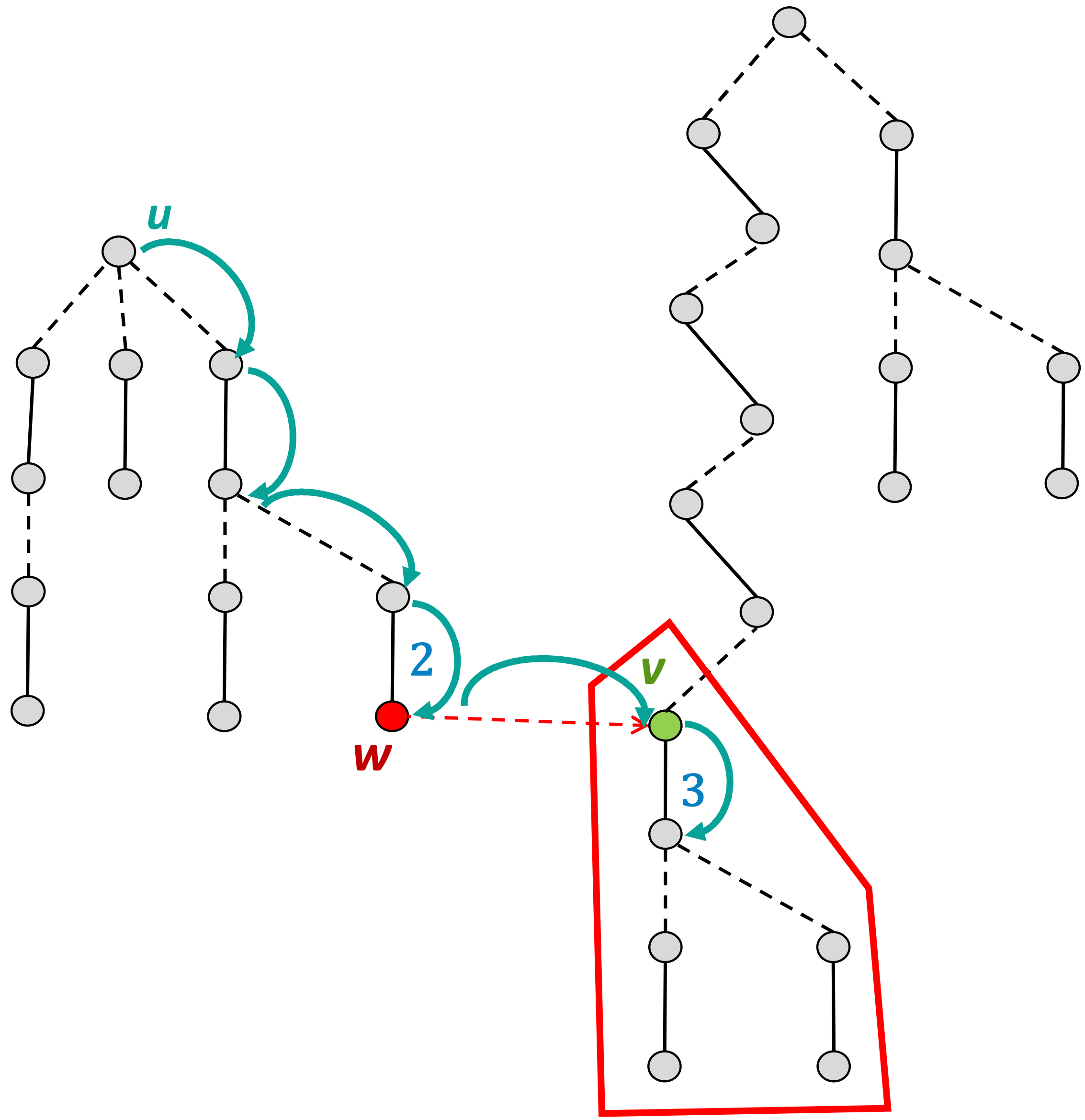


Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree $\rightarrow$ **Extend**

Case 2: ***v*** is an inner vertex $\rightarrow$ **Overtake**
(Also take the subtree of ***v***)

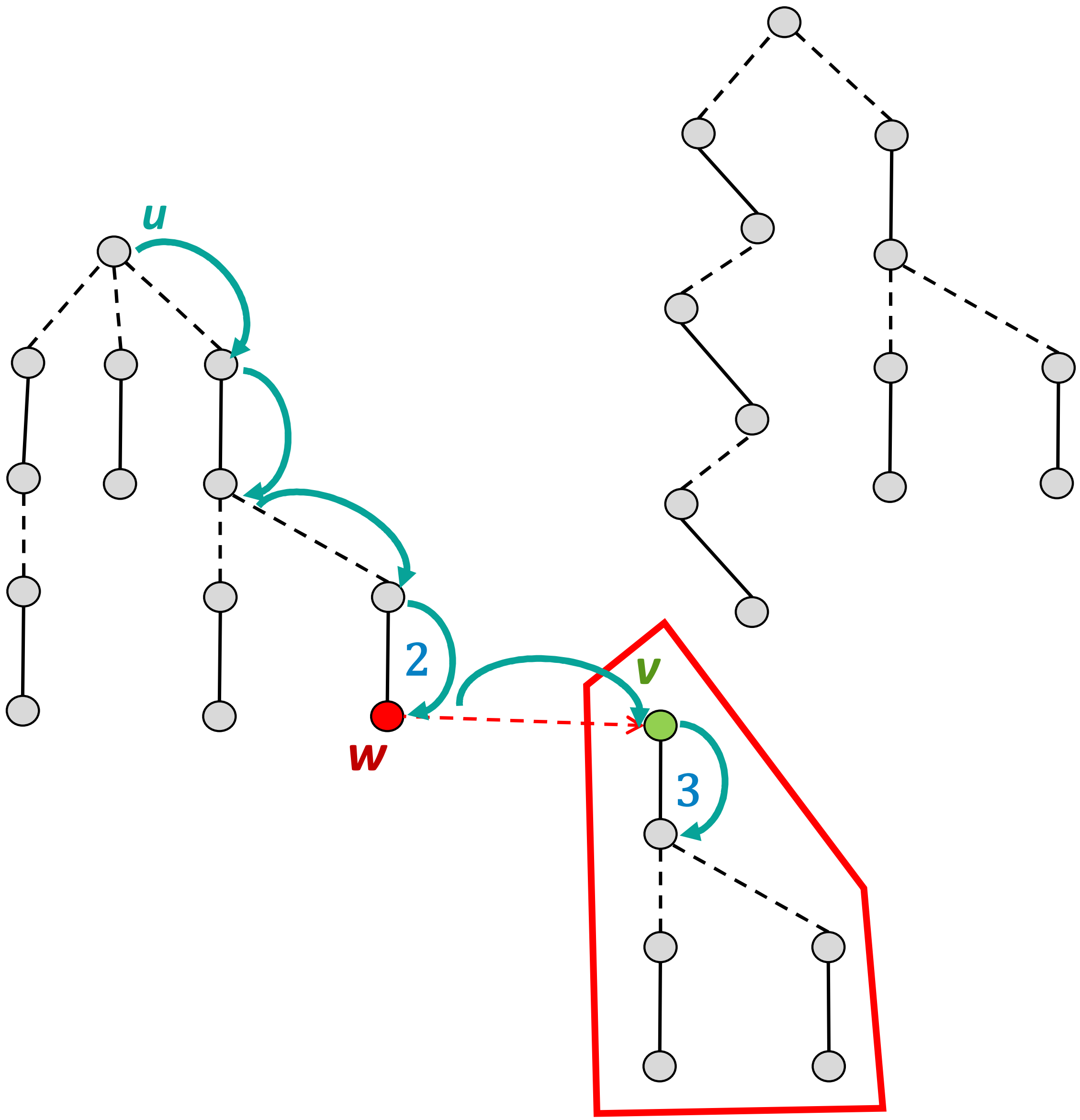


Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)



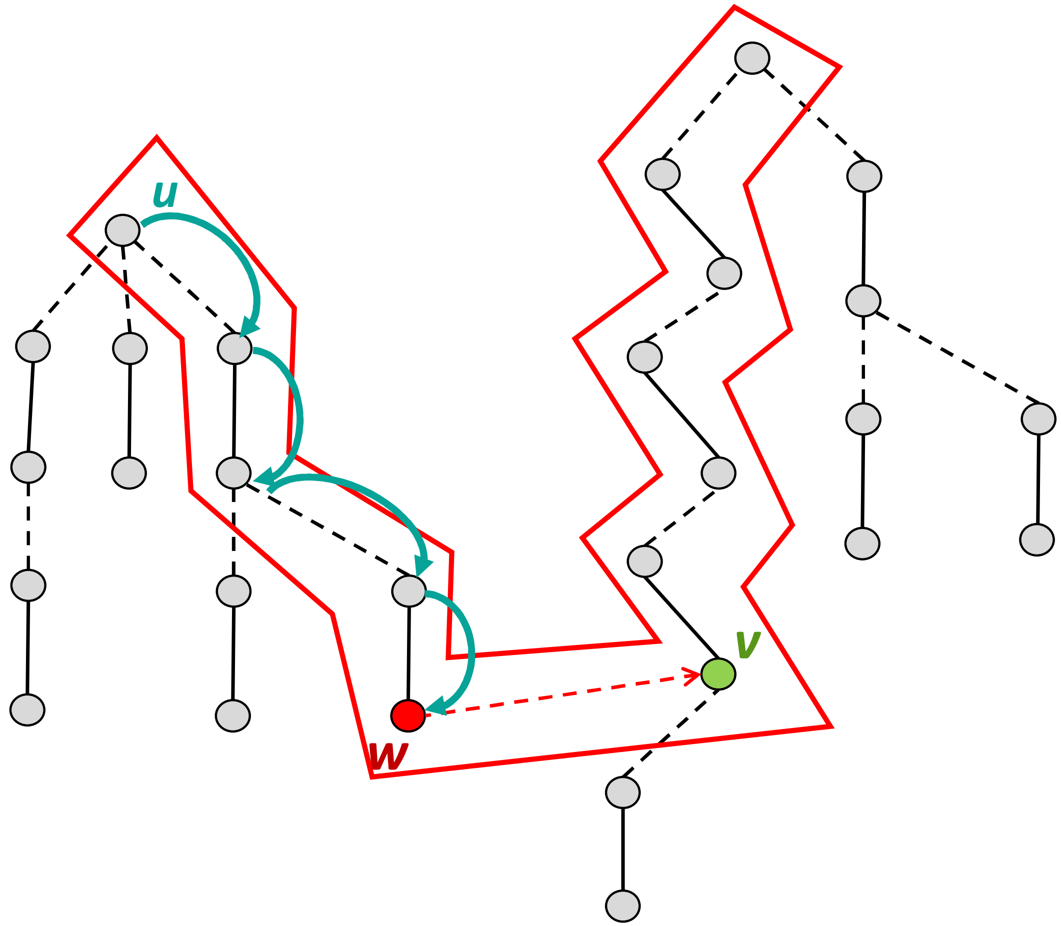
Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)

Case 3: ***v*** is an outer vertex of another tree
→ **Augmentation found**
(remove both trees)



Idea: Maintain alternating trees

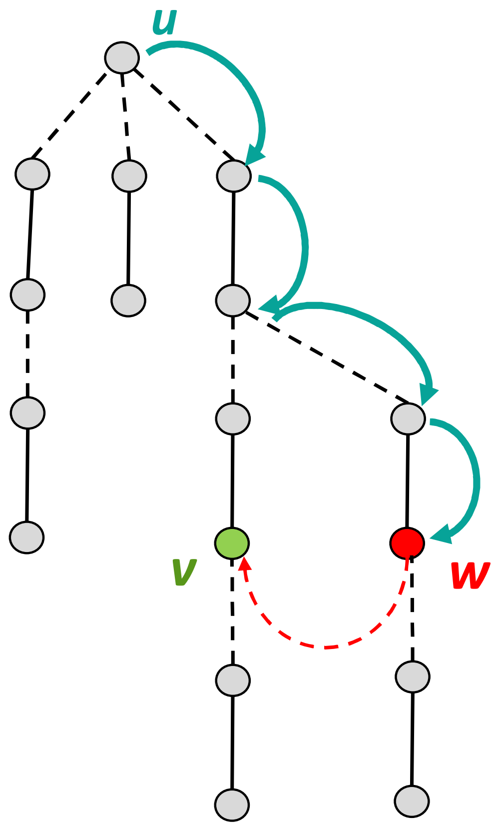
- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)

Case 3: ***v*** is an outer vertex of another tree
→ **Augmentation found**
(remove both trees)

Case 4: ***v*** is an outer vertex of the same tree??
(odd cycle!)



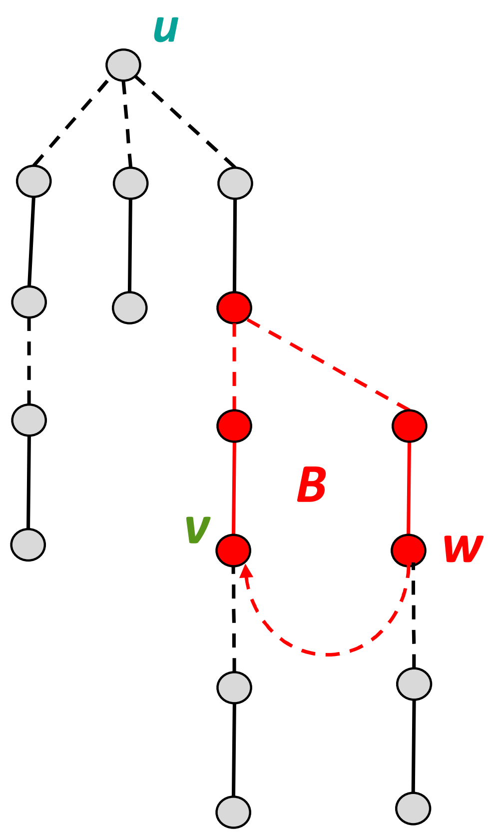
Idea: Blossom contraction

Claim (part 1) [Edmonds, 1965]

Let T be an alternating tree. An edge connecting two outer vertices of T forms a blossom

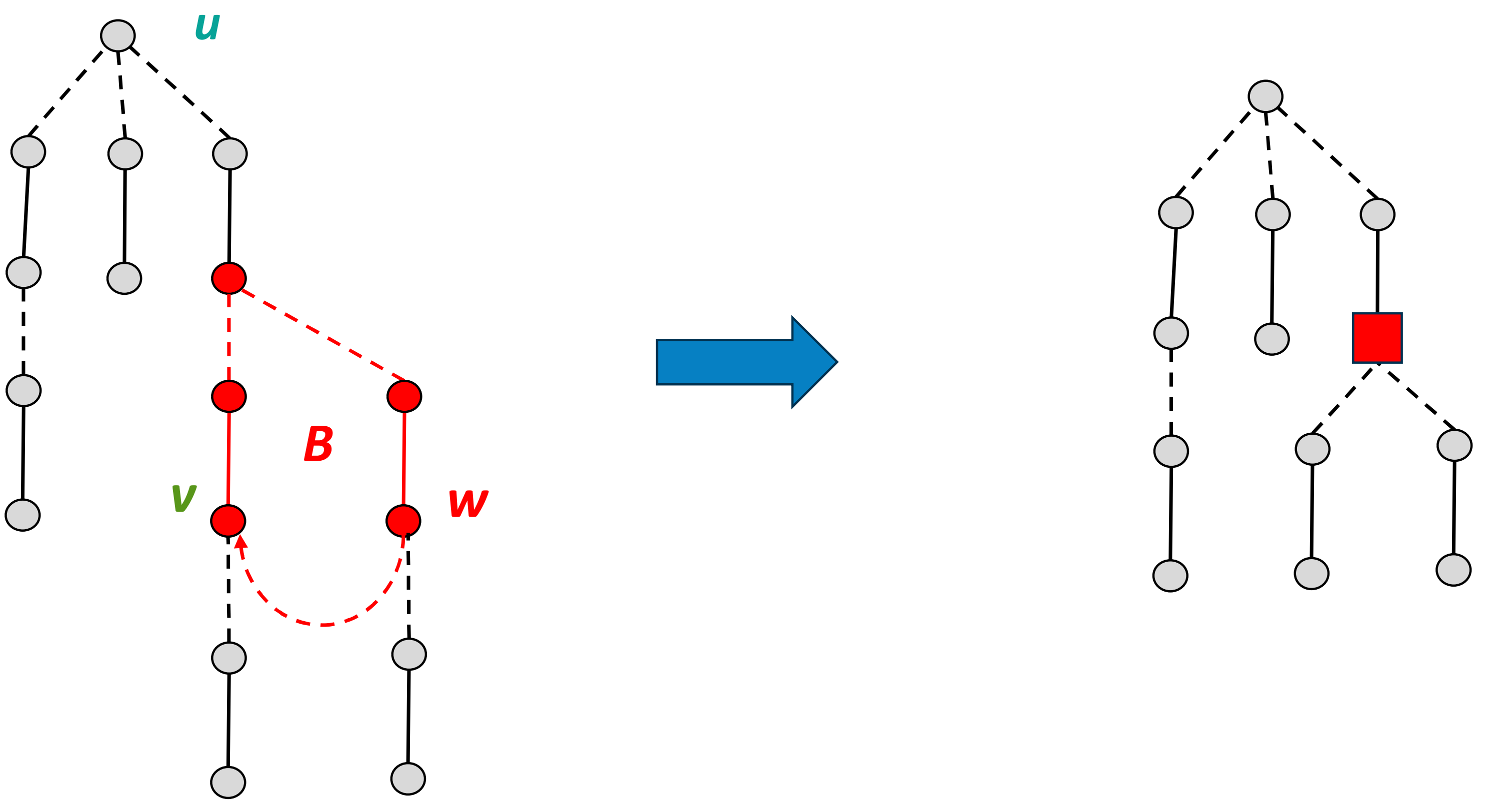
Definition

A blossom is a subgraph that forms an odd cycle with exactly one unmatched vertex



Claim (part 2) [Edmonds, 1965]

By contracting such a blossom, T remains an alternating tree.



Analysis

Find "almost" maximal set of short augmenting paths

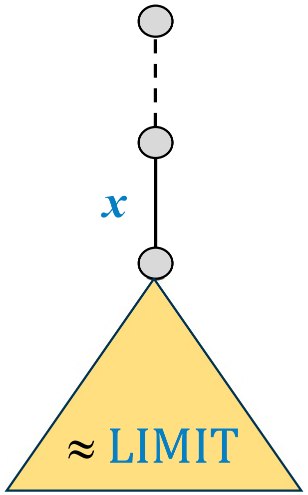
A tree cannot “grow” beyond poly(1/ε)

(Recall that a tree is **removed** after an augmentation.)

A tree cannot "grow" beyond poly($1/\varepsilon$)

Idea: A tree gets frozen when its size reaches $\mathrm{LIMIT} = 1/\varepsilon^2$.

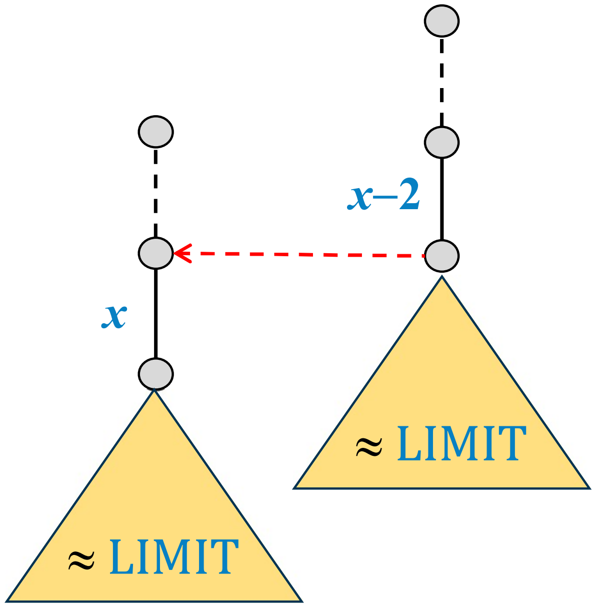
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\mathrm{LIMIT} = 1/\varepsilon^2$.

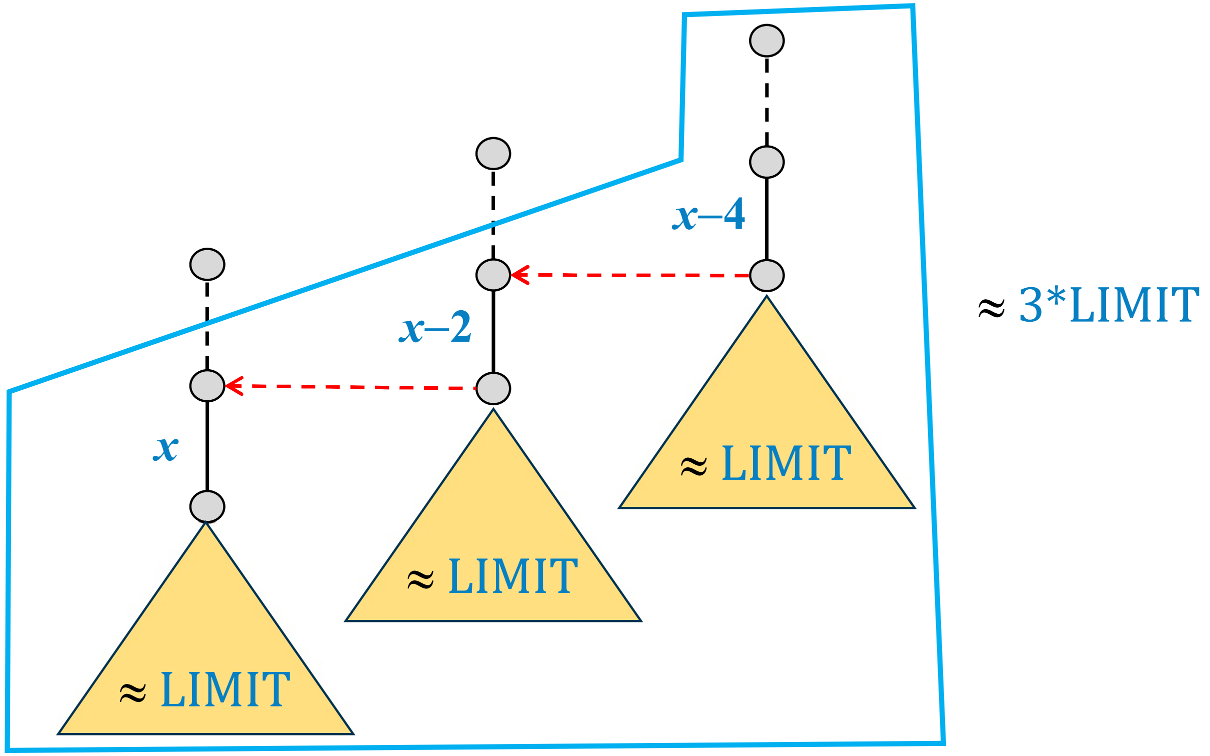
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly($1/\varepsilon$)

Idea: A tree gets frozen when its size reaches LIMIT $= 1/\varepsilon^2$.

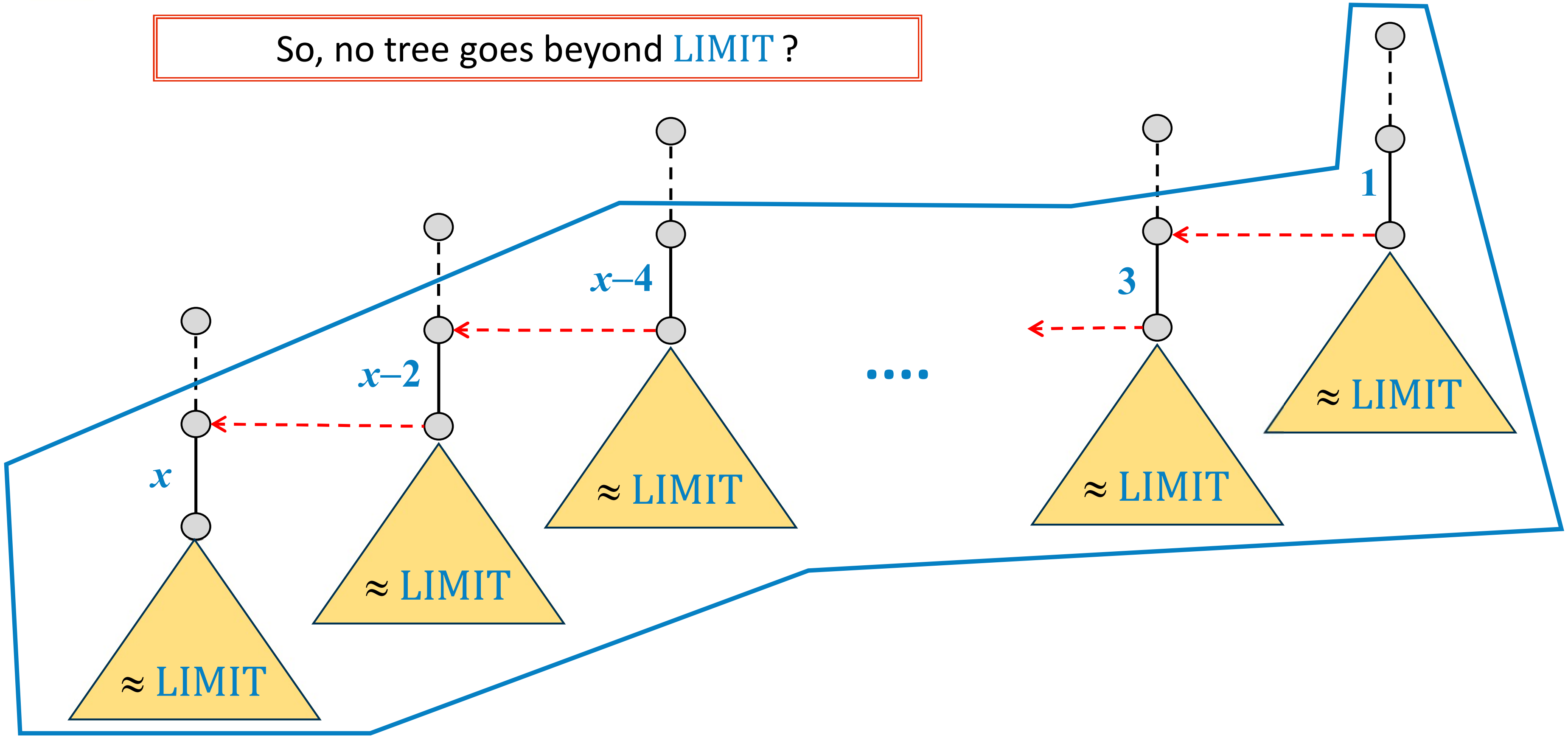
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly($1/\varepsilon$)

Idea: A tree gets frozen when its size reaches LIMIT $= 1/\varepsilon^2$.

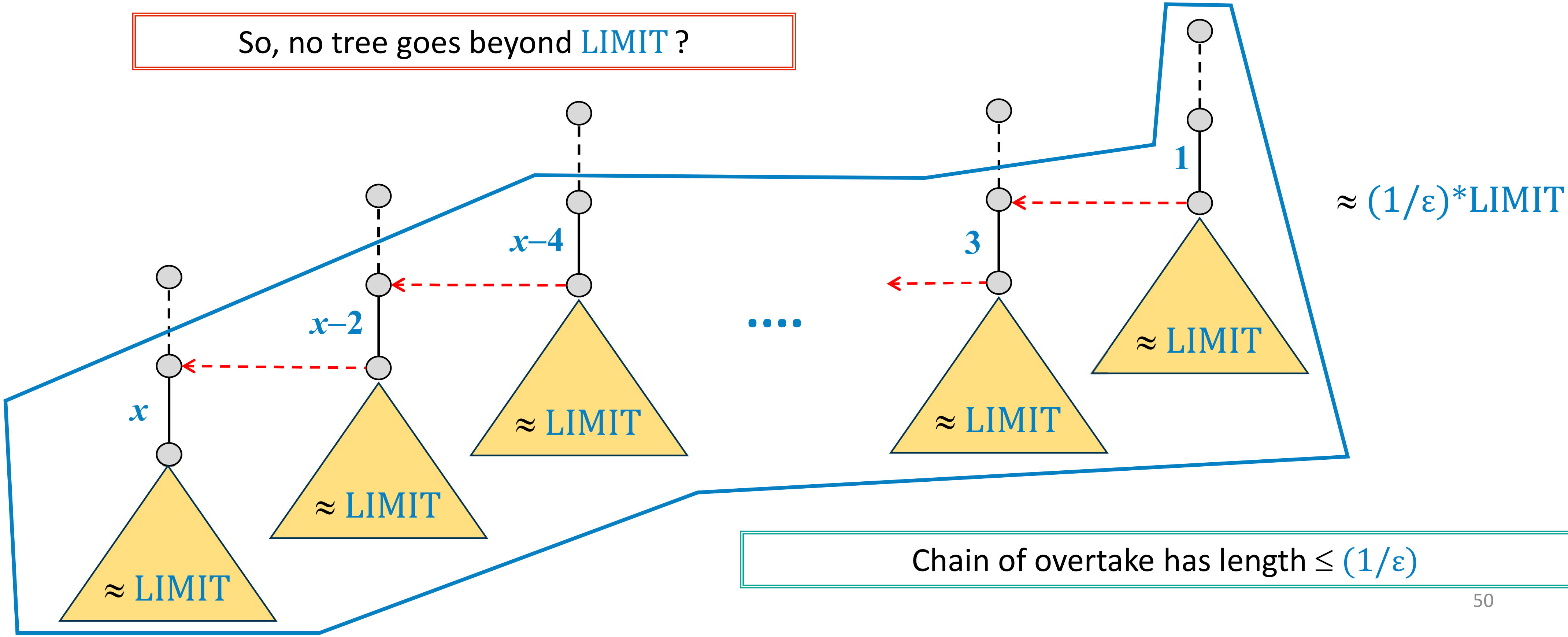
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly($1/\varepsilon$)

Idea: A tree gets frozen when its size reaches LIMIT $= 1/\varepsilon^2$.

So, no tree goes beyond LIMIT ?



Chain of overtake has length $\leq (1/\varepsilon)$

Open questions

Improving the poly-dependence on $1/\varepsilon$.

Further simplification of the current approach

b-matching in $\mathrm{poly}(1/\varepsilon)$ passes in general graphs?