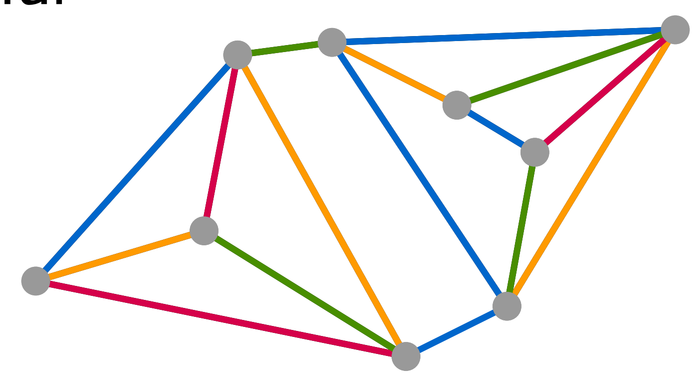
Online Edge Coloring is (Nearly) as Easy as Offline

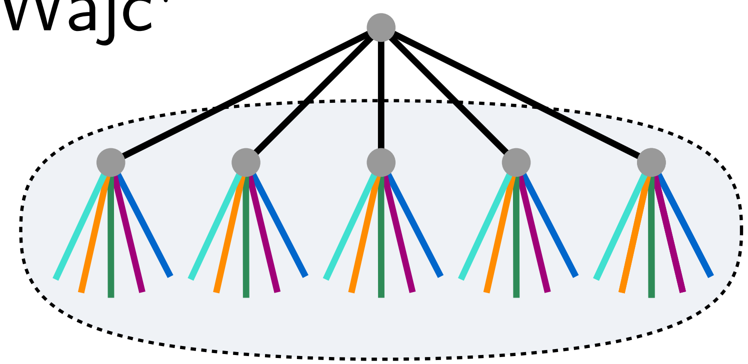
Joakim Blikstad* Ola Svensson†
Radu Vintan† David Wajc‡

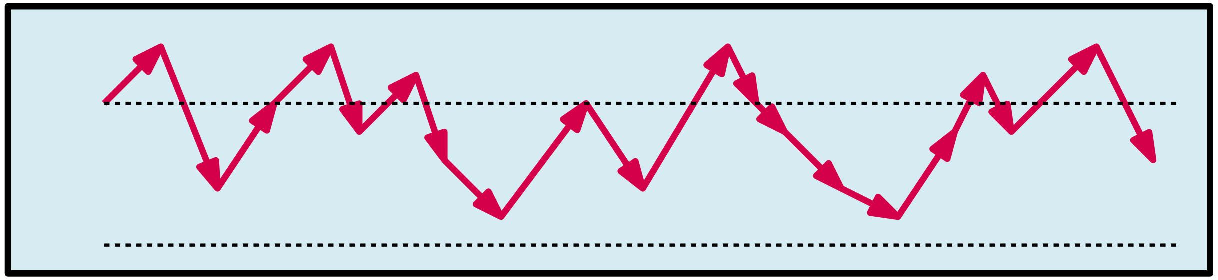
TTIC online seminar

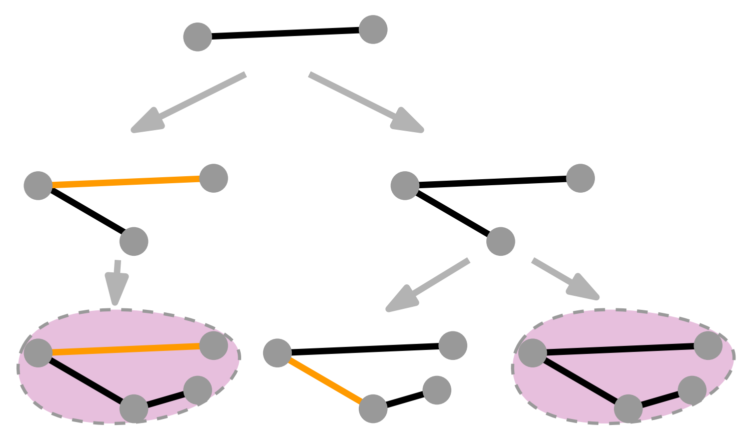
May 2024

*KTH, Sweden & MPI-INF, Germany
†EPFL, Switzerland
‡Technion, Israel



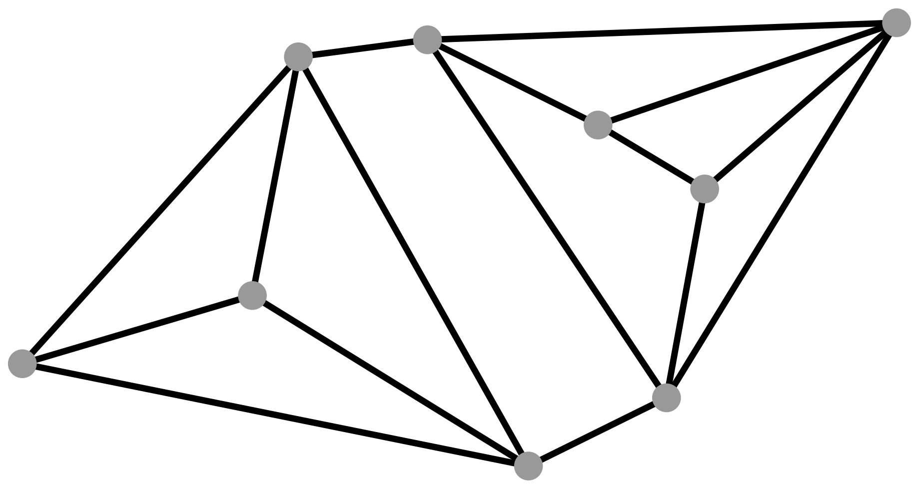






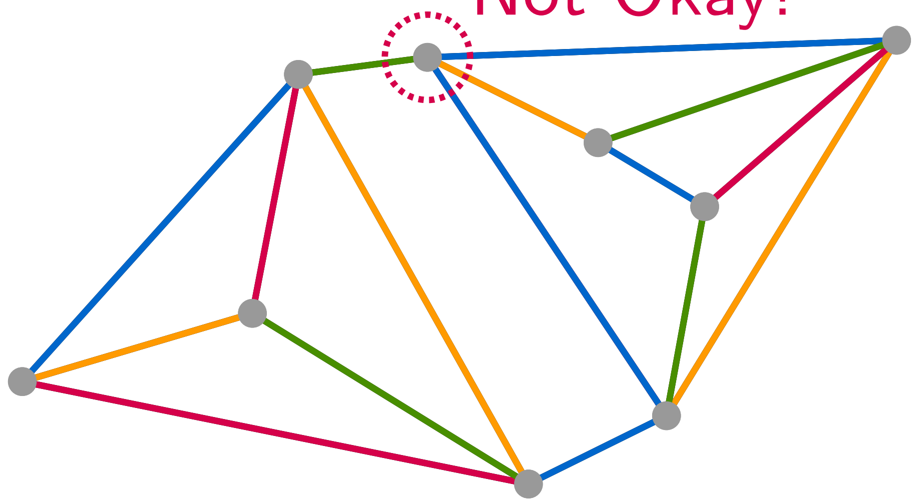
Edge Coloring

Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



Edge Coloring

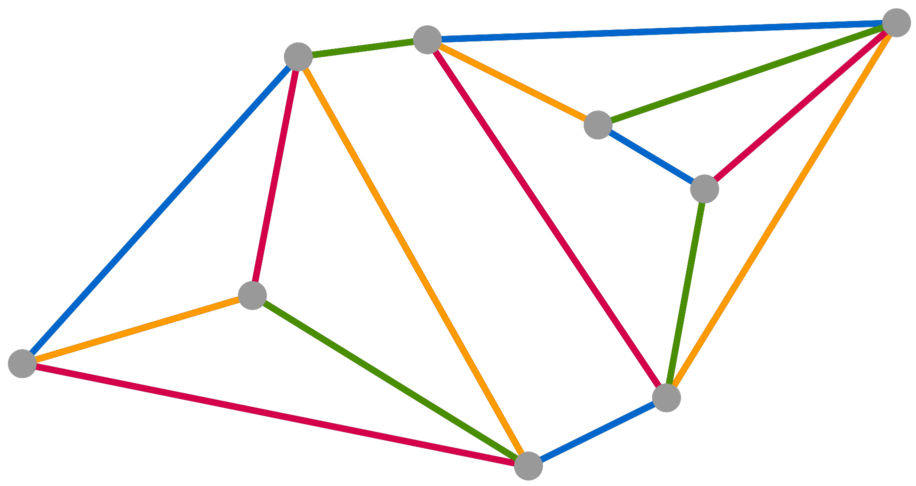
Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



4 colors?

Edge Coloring

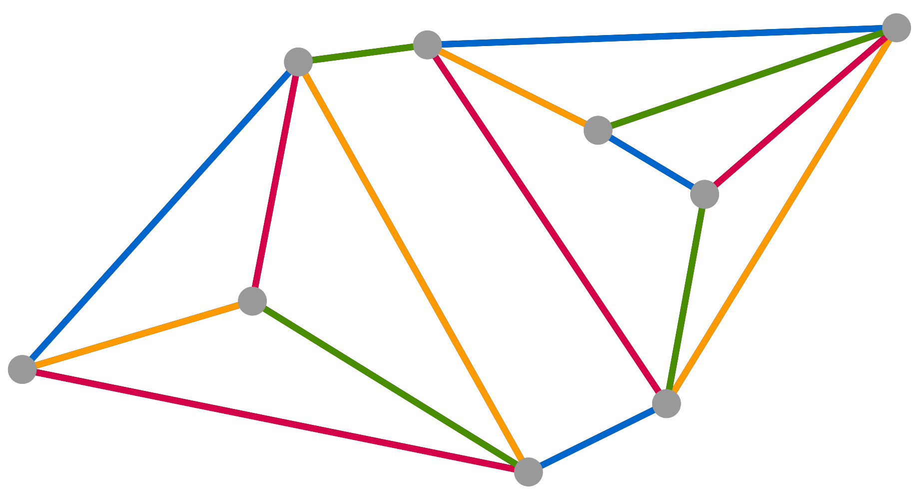
Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



4 colors?

Edge Coloring

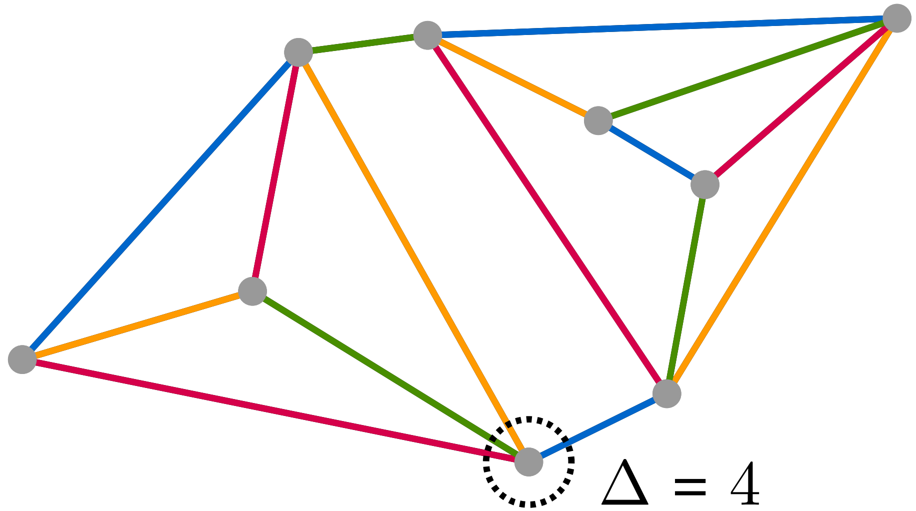
Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



4 colors?
Optimal?

Edge Coloring

Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



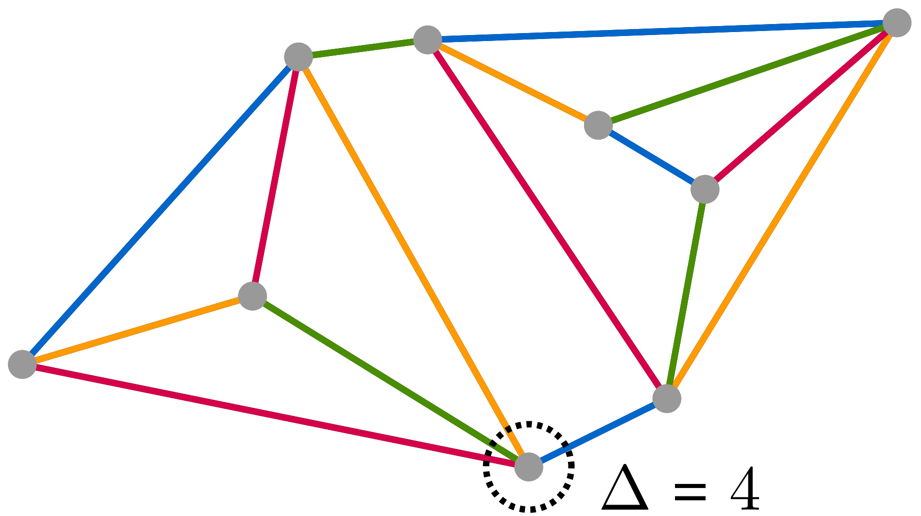
4 colors?
Optimal?

$\Delta := \max_{v \in V} \deg(v)$

Claim: $\#\text{Colors} \geq \Delta$

Edge Coloring

Given: Graph $G = (V, E)$
Goal: Color *edges* with few colors
Constraint: No two incident edges get the same color



4 colors?
Optimal?

$\Delta := \max_{v \in V} \deg(v)$

Claim: $\#\text{Colors} \geq \Delta$

Theorem: $\#\text{Colors} \leq \Delta + 1$

[Vizing 1964]

Edge Coloring Algorithms

- Many algorithms computing $(\Delta + 1)$-edge-colorings
[Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,Misra/Gries'92,...]
- NP-Hard to Δ-edge-color.
[Holyer'81]
- Many algorithms computing Δ-edge-color in *bipartite graphs*
[Cole/Hopcroft'82,Cole/Ost/Schirra'01,Alon'03,Goel/Kapralov/Khanna'13,...]
- Studied in various computational models:

  Distributed [PanconesiSrinivasan'97,DubhashiGrablePanconessi'98,...]
  PRAM [LevPippengerValiant'81,...]
  NC & RNC [KarloffShmoys'87, MotwaniNaorNaor'94,...]
  Dynamic [Bhattacharya/Chakrabarty/Henzinger/Nanongkai'18,...]
  ...

Edge Coloring Algorithms

- Many algorithms computing $(\Delta + 1)$-edge-colorings
[Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,Misra/Gries'92,...]

- NP-Hard to Δ-edge-color.
[Holyer'81]

- Many algorithms computing Δ-edge-color in *bipartite graphs*
[Cole/Hopcroft'82,Cole/Ost/Schirra'01,Alon'03,Goel/Kapralov/Khanna'13,...]

- Studied in various computational models:

  Distributed [PanconesiSrinivasan'97,DubhashiGrablePanconessi'98,...]
  PRAM [LevPippengerValiant'81,...]
  NC & RNC [KarloffShmoys'87, MotwaniNaorNaor'94,...]
  Dynamic [Bhattacharya/Chakrabarty/Henzinger/Nanongkai'18,...]
  ...

This Talk: Online

Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.

Online Edge Coloring

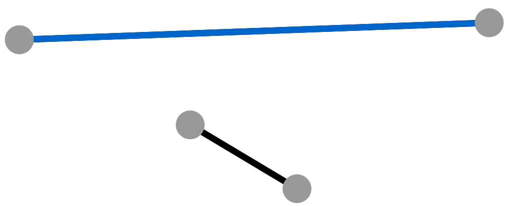
Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.

Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.

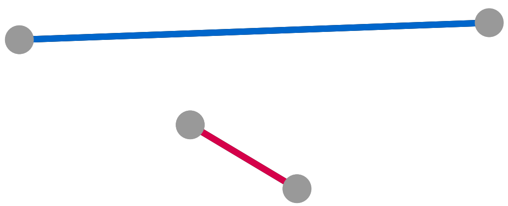
Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



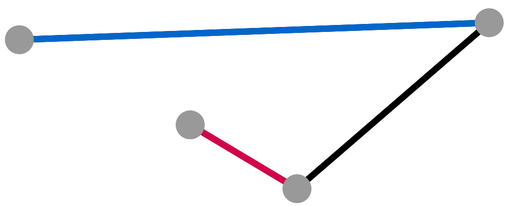
Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



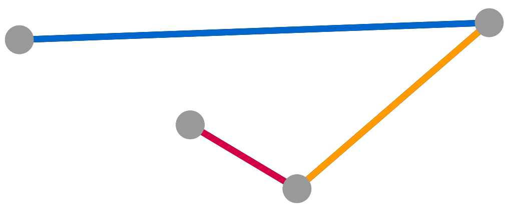
Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



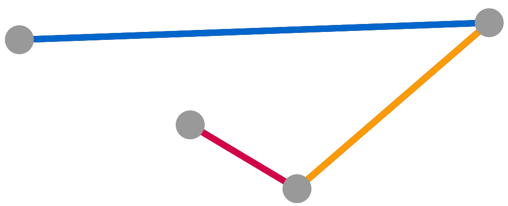
Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



Online Edge Coloring

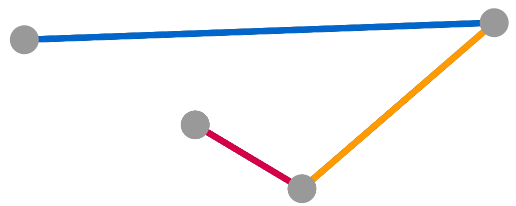
Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



Variants:
Edge or Vertex arrivals
Adversarial or Random order
Deterministic or Oblivious or Adaptive
General or Bipartite graphs

Online Edge Coloring

Online: Graph revealed over time. Max-degree Δ known.
Task: Color edge *irrevocably* when it is revealed.



Variants:
Edge or Vertex arrivals
Adversarial or Random order
Deterministic or Oblivious or Adaptive
General or Bipartite graphs

How many colors do we need? Still $\approx \Delta$?

Warm-up: Greedy Algorithm

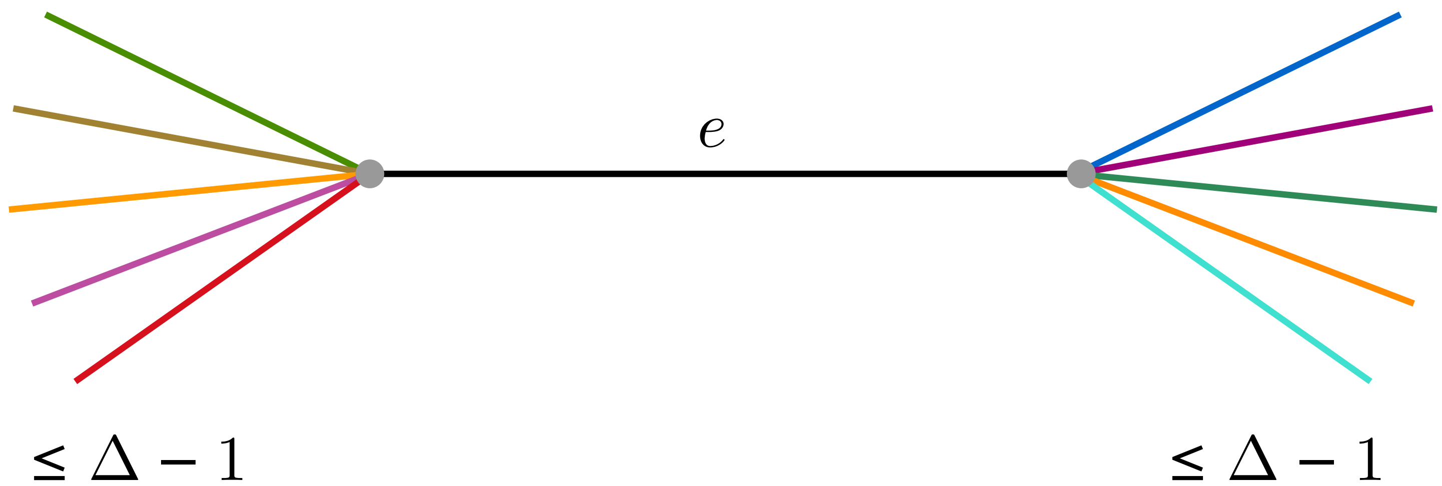
Greedy: Color edge with "lowest" avaliable color.

$$\text{Colors} = \{1, 2, 3, \ldots\}$$

Warm-up: Greedy Algorithm

Greedy: Color edge with "lowest" avaliable color.

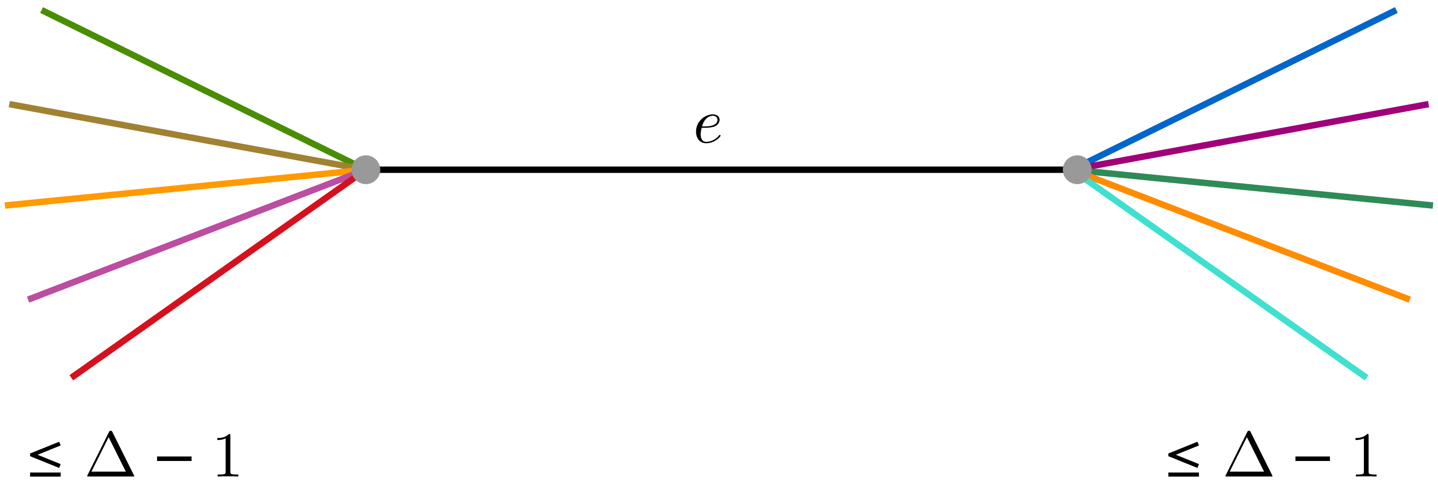
Colors $= \{1, 2, 3, \ldots\}$



Warm-up: Greedy Algorithm

Greedy: Color edge with "lowest" avaliable color.

Colors $= \{1, 2, 3, \ldots\}$



Claim: $\leq 2(\Delta - 1)$ blocked colors

Claim: Greedy uses $\leq 2\Delta - 1$ colors

Can we do better?

Can we beat $2\Delta - 1$ colors?

Can we do better?

Can we beat $2\Delta - 1$ colors?

NO!

Lower Bound

Theorem: No online algorithm can $(2\Delta - 2)$-edge-color every graph.

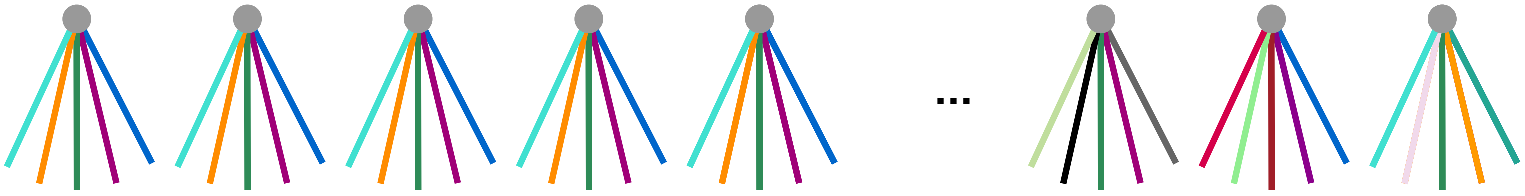
[Bar-Noy/Motwani/Naor 1992]

Lower Bound

Theorem: No online algorithm can $(2\Delta - 2)$-edge-color every graph.

[Bar-Noy/Motwani/Naor 1992]

Idea: Create lots of $(\Delta - 1)$-stars



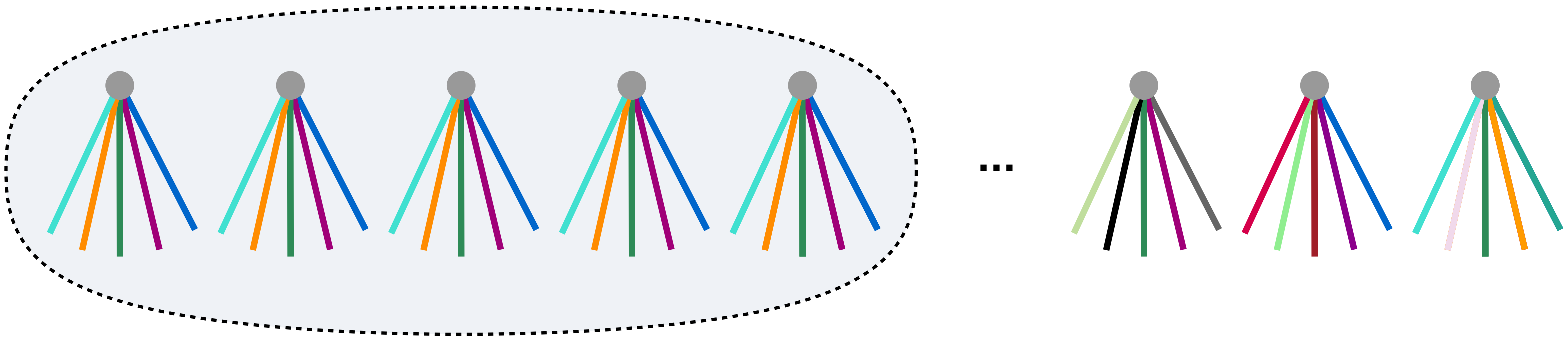
Lower Bound

Theorem: No online algorithm can $(2\Delta - 2)$-edge-color every graph.

[Bar-Noy/Motwani/Naor 1992]

Idea: Create lots of $(\Delta - 1)$-stars

Eventually have Δ stars colored the same (pigeonhole principle)



Lower Bound

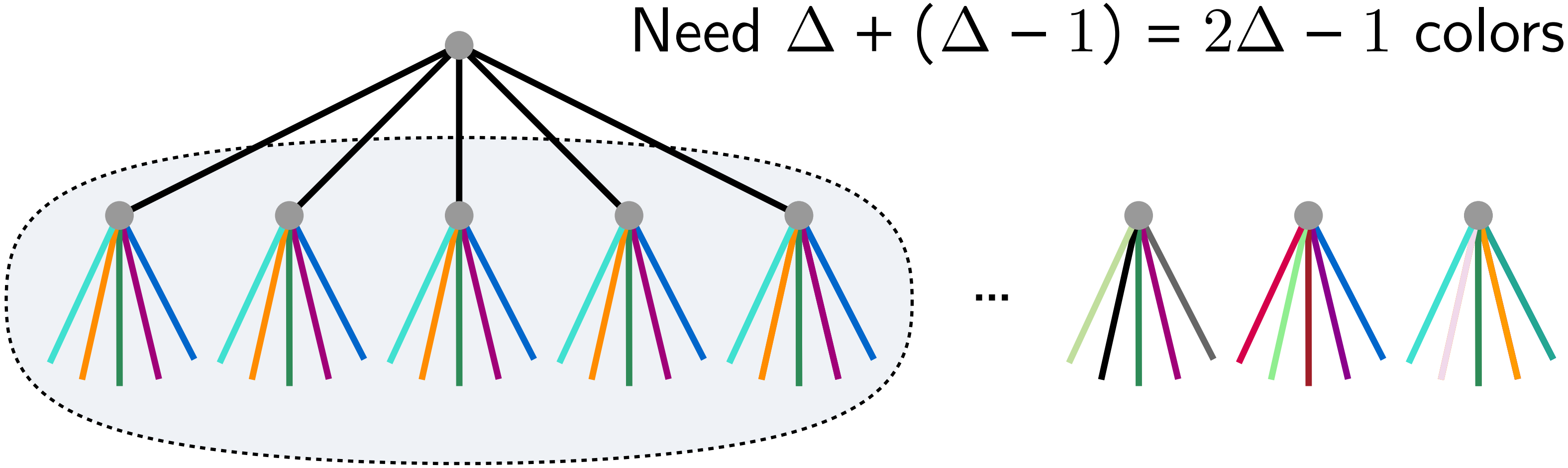
Theorem: No online algorithm can $(2\Delta - 2)$-edge-color every graph.

[Bar-Noy/Motwani/Naor 1992]

Idea: Create lots of $(\Delta - 1)$-stars

Eventually have Δ stars colored the same (pigeonhole principle)

Need $\Delta + (\Delta - 1) = 2\Delta - 1$ colors



Lower Bound

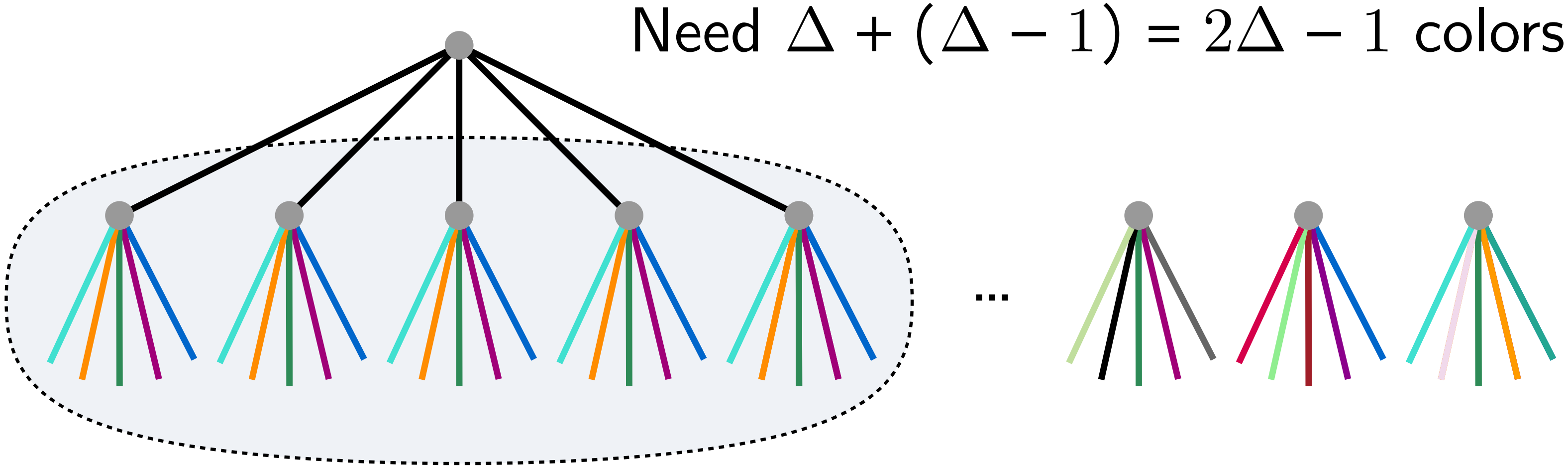
Theorem: No online algorithm can $(2\Delta - 2)$-edge-color every graph.

[Bar-Noy/Motwani/Naor 1992]

Idea: Create lots of $(\Delta - 1)$-stars

Eventually have Δ stars colored the same (pigeonhole principle)

Need $\Delta + (\Delta - 1) = 2\Delta - 1$ colors



Can we do better?

Can we beat $2\Delta - 1$ colors?

NO!

Can we do better?

Can we beat $2\Delta - 1$ colors?

NO!

Silver Lining: LB requires $\Delta \cdot \binom{2\Delta-2}{\Delta-1} \approx 4^{\Delta}$ stars, that is $\Delta = O(\log n)$

Can we do better?

Can we beat $2\Delta - 1$ colors?

NO!

Silver Lining: LB requires $\Delta \cdot \binom{2\Delta-2}{\Delta-1} \approx 4^{\Delta}$ stars, that is $\Delta = O(\log n)$

Can we do better when $\Delta = \omega(\log n)$?

Can we do better?

Can we beat $2\Delta - 1$ colors?

NO!

Silver Lining: LB requires $\Delta \cdot \binom{2\Delta-2}{\Delta-1} \approx 4^{\Delta}$ stars, that is $\Delta = O(\log n)$

Can we do better when $\Delta = \omega(\log n)$?

YES $\approx \Delta$ colors :)

Progress

Conjecture: $(1 + o(1))\Delta$-colors sufficent when $\Delta = \omega(\log n)$.

[Bar-Noy/Motwani/Naor 1992]

Progress

Conjecture: $(1 + o(1))\Delta$-colors sufficent when $\Delta = \omega(\log n)$.

[Bar-Noy/Motwani/Naor 1992]

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$-coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\mathbf{1.27\Delta}$-coloring if $\Delta = \omega(\log n)$
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$-coloring if $\Delta = \omega(\log n)$
- Adversarial vertex arrivals:
  - [Cohen/Peng/Wajc'19] (simplified [B./Svensson/Vintan/Wajc'24]:
    $\approx \Delta$-coloring **bipartite graphs**

    For unknown Δ: $\approx \frac{e}{e-1}\Delta$-coloring **bipartite graphs (optimal)**
  - [Saberi/Wajc'21]: $\approx 1.9\Delta$-coloring **general graphs**
- Adversarial edge arrivals
  - [Kulkarni/Liu/Sah/Sawhney/Tarnawski'22] $\approx \frac{e}{e-1}\Delta$-coloring

Progress

Conjecture: $(1+o(1))\Delta$-colors sufficent when $\Delta = \omega(\log n)$.

[Bar-Noy/Motwani/Naor 1992]

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$-coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\mathbf{1.27\Delta}$-coloring if $\Delta = \omega(\log n)$
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$-coloring if $\Delta = \omega(\log n)$
- Adversarial vertex arrivals:
  - [Cohen/Peng/Wajc'19] (simplified [B./Svensson/Vintan/Wajc'24]:
    $\approx \Delta$-coloring **bipartite graphs**

    For unknown Δ: $\approx \frac{e}{e-1}\Delta$-coloring **bipartite graphs (optimal)**
  - [Saberi/Wajc'21]: $\approx 1.9\Delta$-coloring **general graphs**
- Adversarial edge arrivals
  - [Kulkarni/Liu/Sah/Sawhney/Tarnawski'22] $\approx \frac{e}{e-1}\Delta$-coloring

This Talk: $\approx \Delta$ colors, most general setting of advesarial edge arrivals

Progress

Theorem:
~~**Conjecture:**~~ $(1 + o(1))\Delta$-colors sufficent when $\Delta = \omega(\log n)$.

[Bar-Noy/Motwani/Naor 1992]

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$-coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\mathbf{1.27\Delta}$-coloring if $\Delta = \omega(\log n)$
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$-coloring if $\Delta = \omega(\log n)$
- Adversarial vertex arrivals:
  - [Cohen/Peng/Wajc'19] (simplified [B./Svensson/Vintan/Wajc'24]: $\approx \Delta$-coloring **bipartite graphs**

    For unknown Δ: $\approx \frac{e}{e-1}\Delta$-coloring **bipartite graphs (optimal)**
  - [Saberi/Wajc'21]: $\approx 1.9\Delta$-coloring **general graphs**
- Adversarial edge arrivals
  - [Kulkarni/Liu/Sah/Sawhney/Tarnawski'22] $\approx \frac{e}{e-1}\Delta$-coloring

This Talk: $\approx \Delta$ colors, most general setting of advesarial edge arrivals

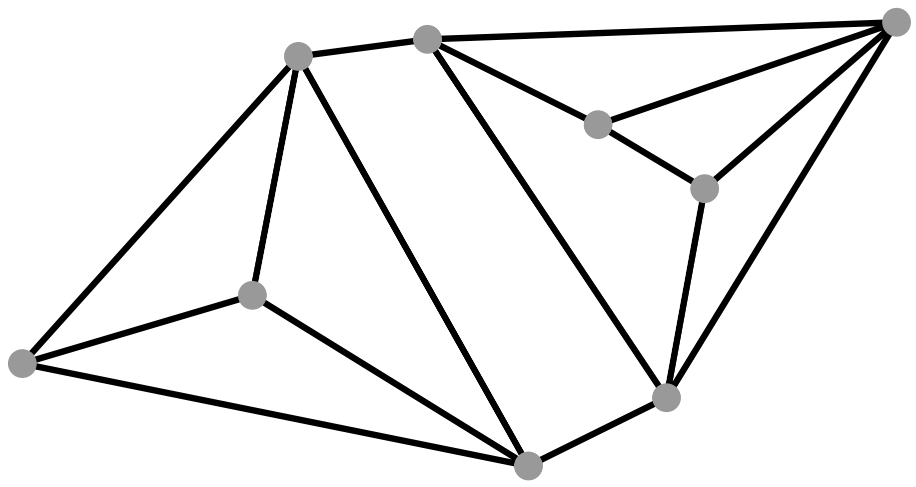
Techniques

Technical Part — Outline

- Edge Coloring $\iff$ Fair Matchings
  - Reduction
- Online Fair Matching Algorithm
  - First Attempt
  - New Algorithm
  - Analysis: Martingales

Fair Matching Problem

Given: Graph $G = (V, E)$
Goal: Find a matching M
α-**Fairness:** $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for each edge $e \in E$

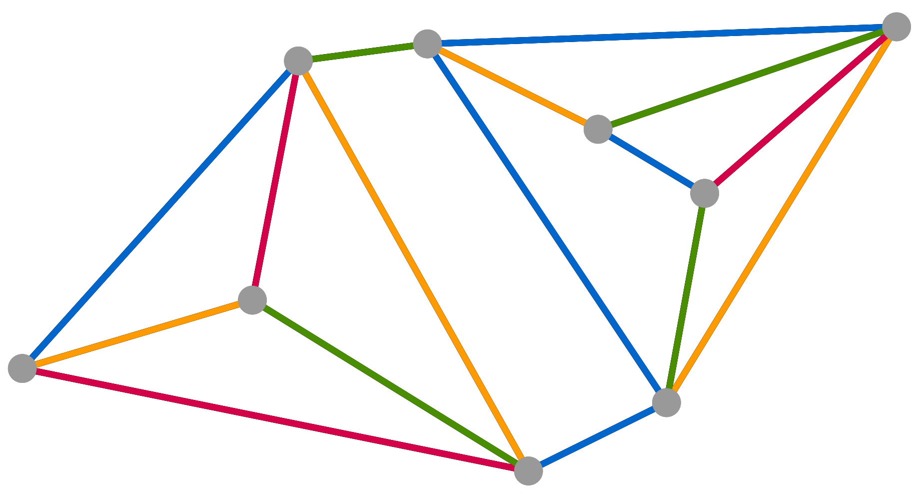


Fair Matching Problem

Given: Graph $G = (V, E)$
Goal: Find a matching M
α-**Fairness:** $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for each edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm

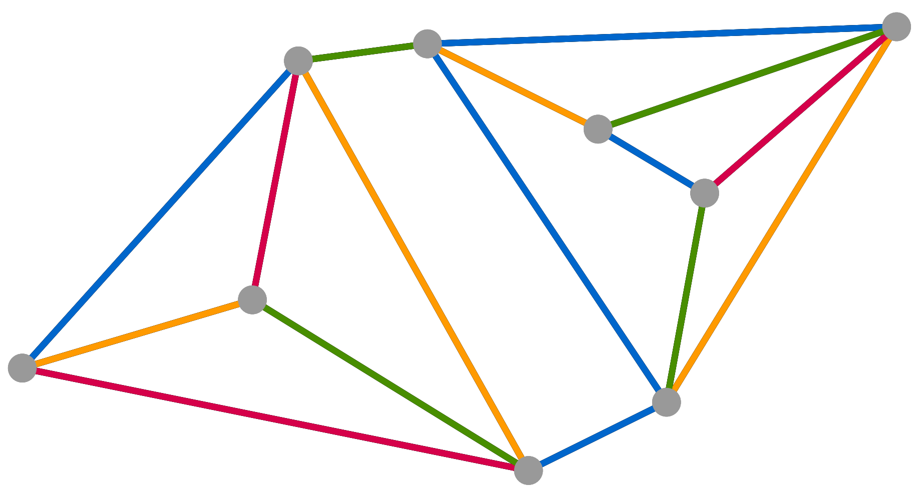
Proof: Pick random color as matching



Fair Matching Problem

Given: Graph $G = (V, E)$
Goal: Find a matching M
α-**Fairness:** $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for each edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm
Proof: Pick random color as matching



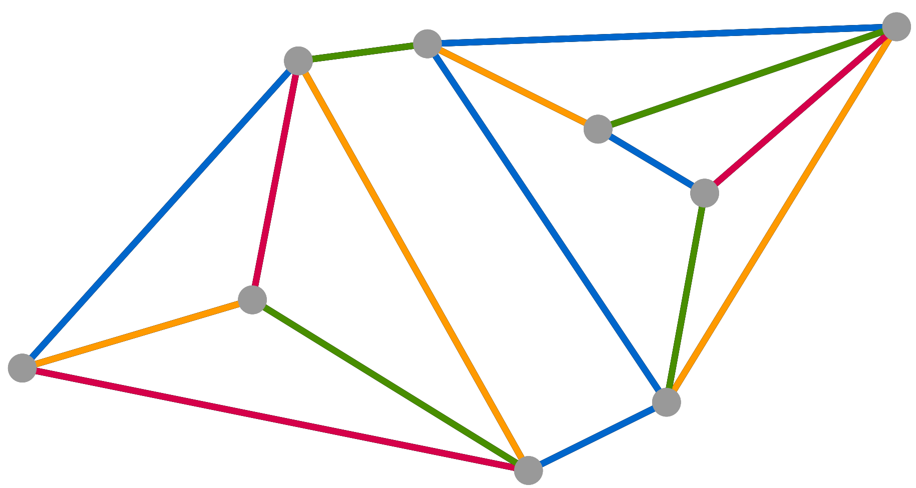
Lemma: α-fair matching $\implies$ $(1 + o(1))\alpha\Delta$-edge-coloring

[Cohen/Peng/Wajc'19]

Fair Matching Problem

Given: Graph $G = (V, E)$
Goal: Find a matching M
α-**Fairness:** $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for each edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm
Proof: Pick random color as matching



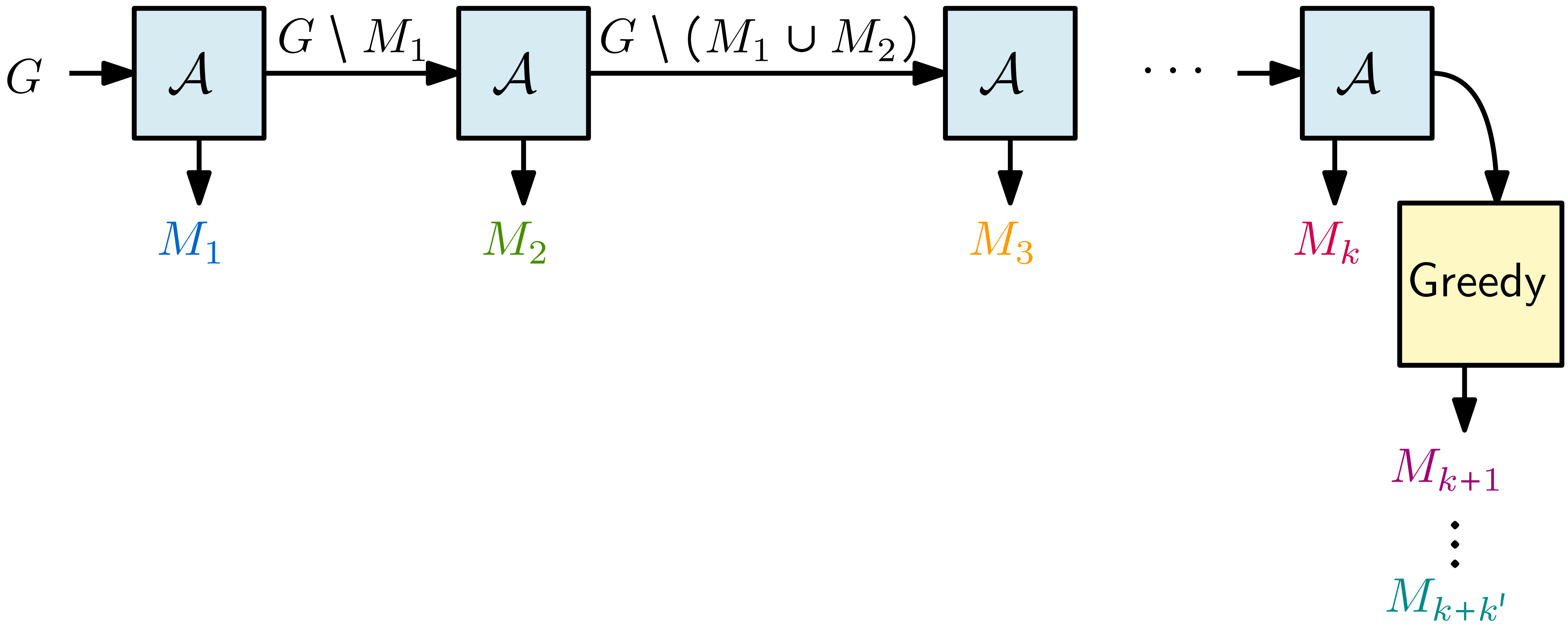
Lemma: α-fair matching $\implies$ $(1 + o(1))\alpha\Delta$-edge-coloring

[Cohen/Peng/Wajc'19]

New Objective: $(1 + o(1))$-**fair matching algorithm**

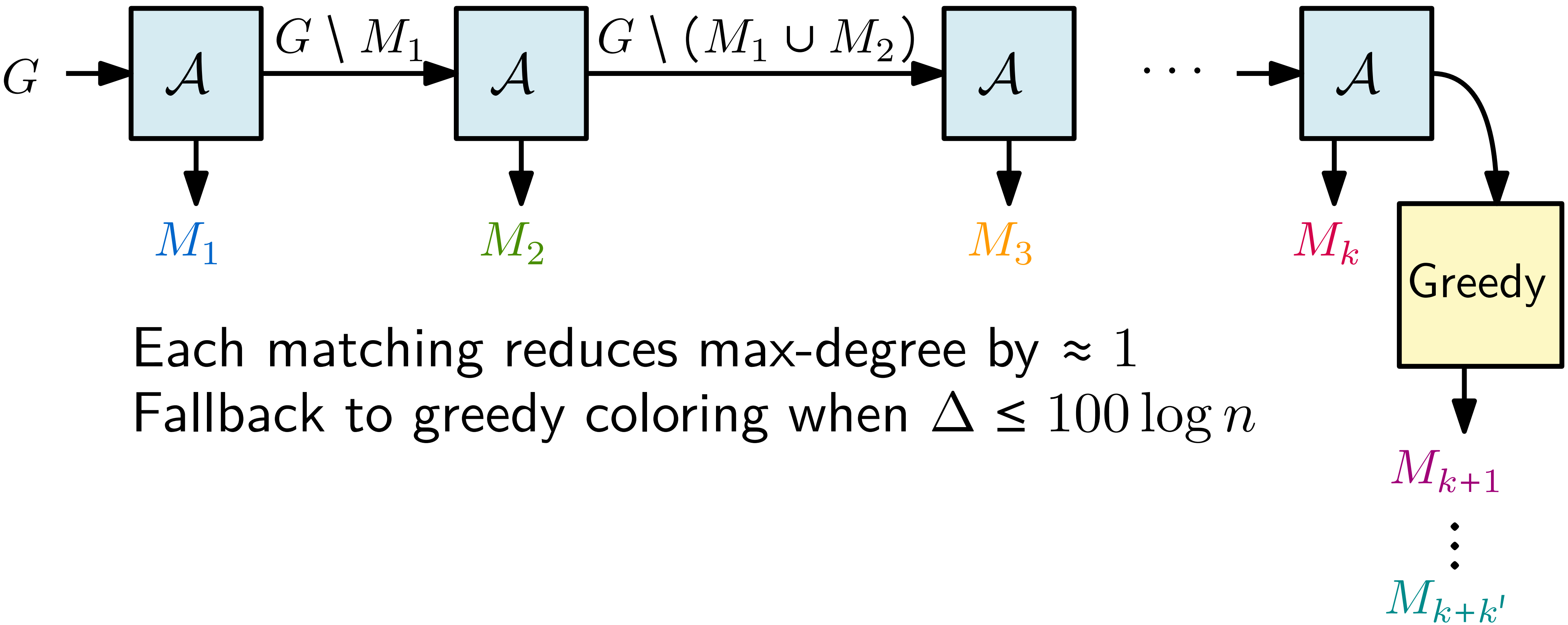
From Fair Matchings to Edge Coloring [Cohen/Peng/Wajc'19]

$\mathcal{A}$: $(1 + o(1))$-fair matching algorithm



From Fair Matchings to Edge Coloring [Cohen/Peng/Wajc'19]

$\mathcal{A}$: $(1 + o(1))$-fair matching algorithm



Each matching reduces max-degree by ≈ 1

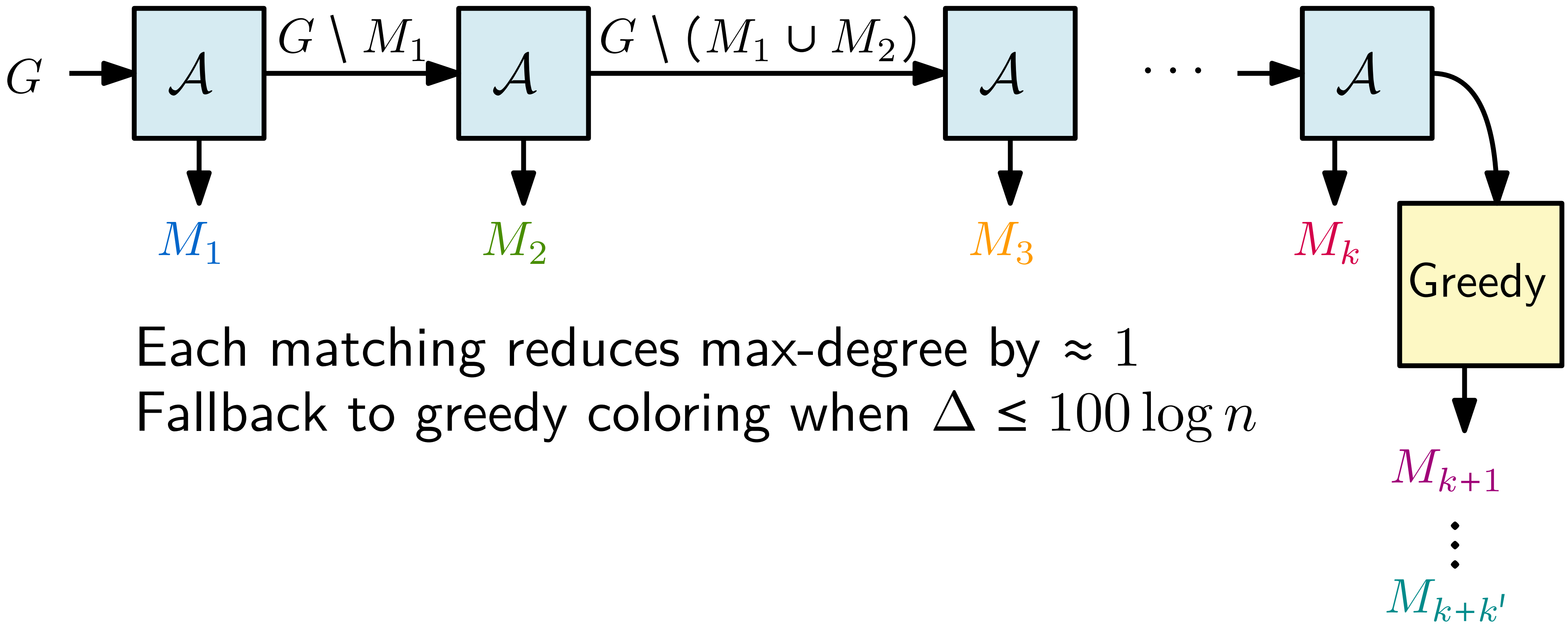
Fallback to greedy coloring when $\Delta \leq 100 \log n$

From Fair Matchings to Edge Coloring [Cohen/Peng/Wajc'19]

$\mathcal{A}$: $(1 + o(1))$-fair matching algorithm

Lemma: α-fair matching $\implies$ $(1 + o(1))\alpha\Delta$-edge-coloring

New Objective: $(1 + o(1))$-fair matching algorithm



Each matching reduces max-degree by ≈ 1

Fallback to greedy coloring when $\Delta \leq 100 \log n$

Fair Matching Algorithm

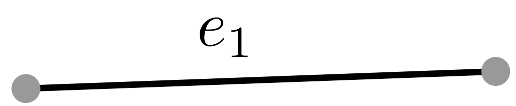
Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

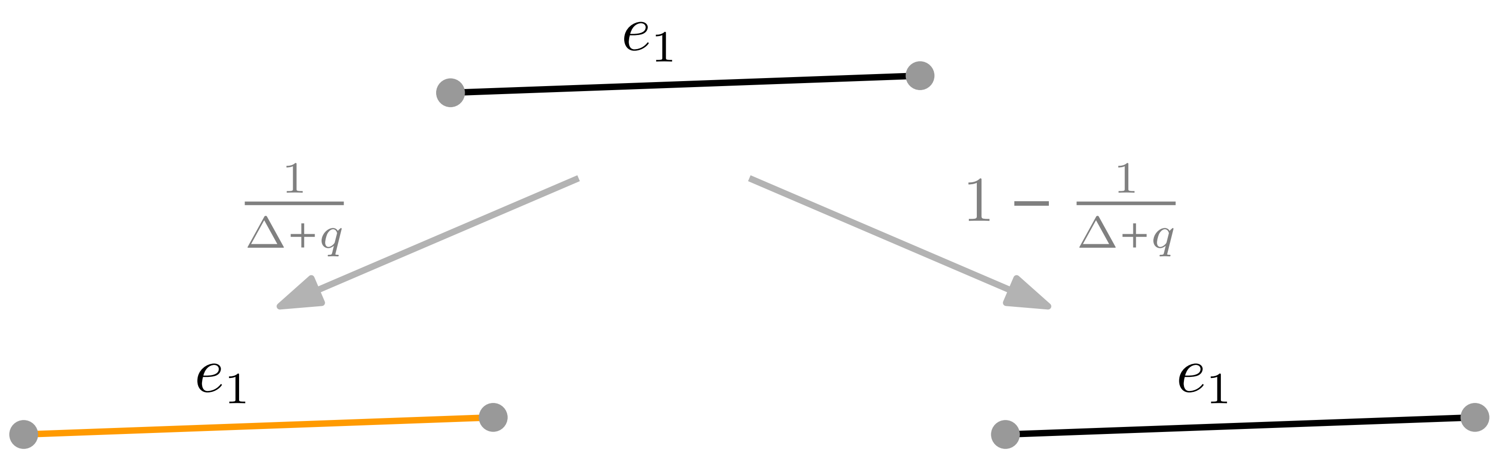
$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

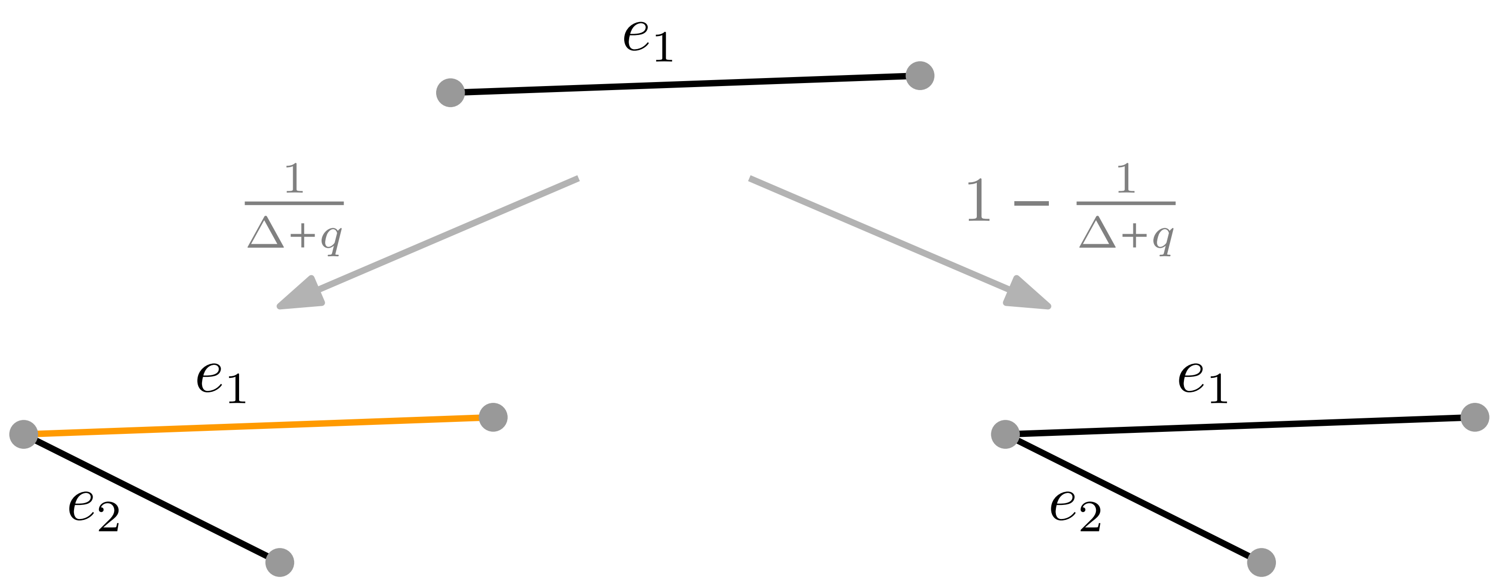
$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

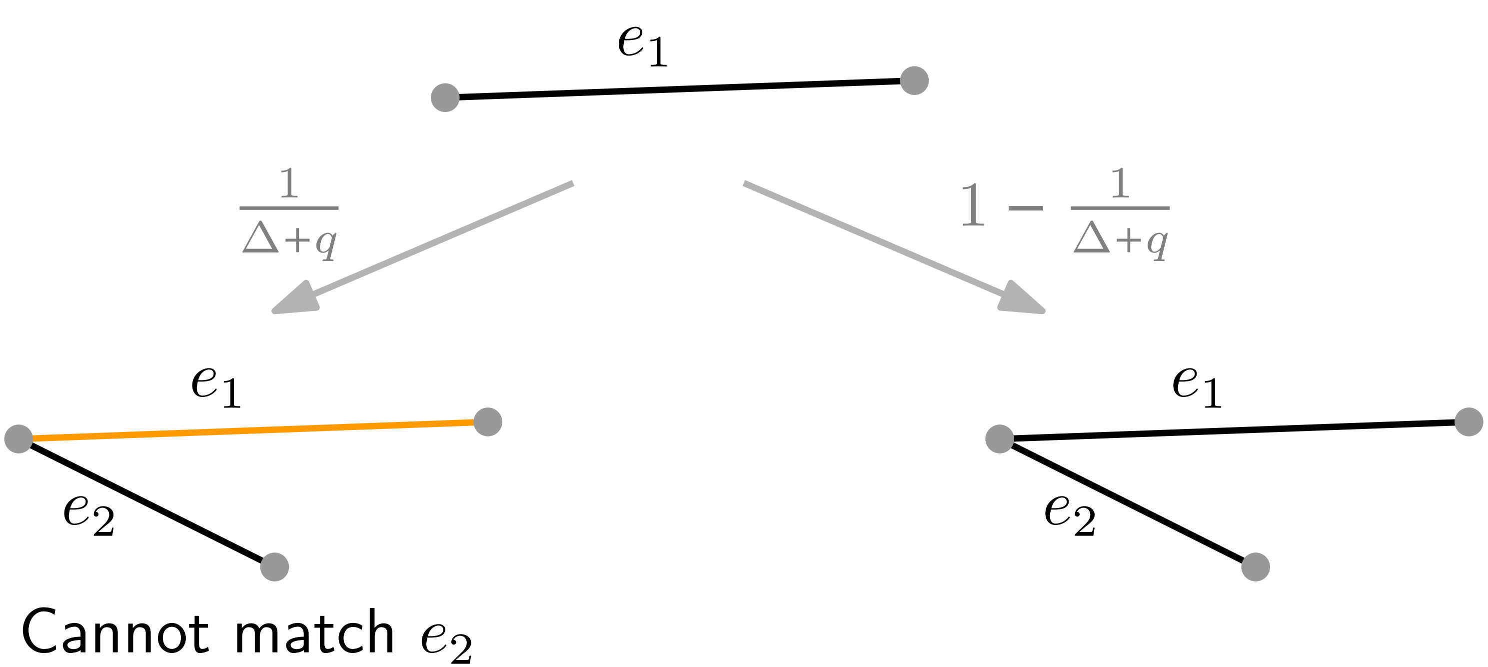
$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

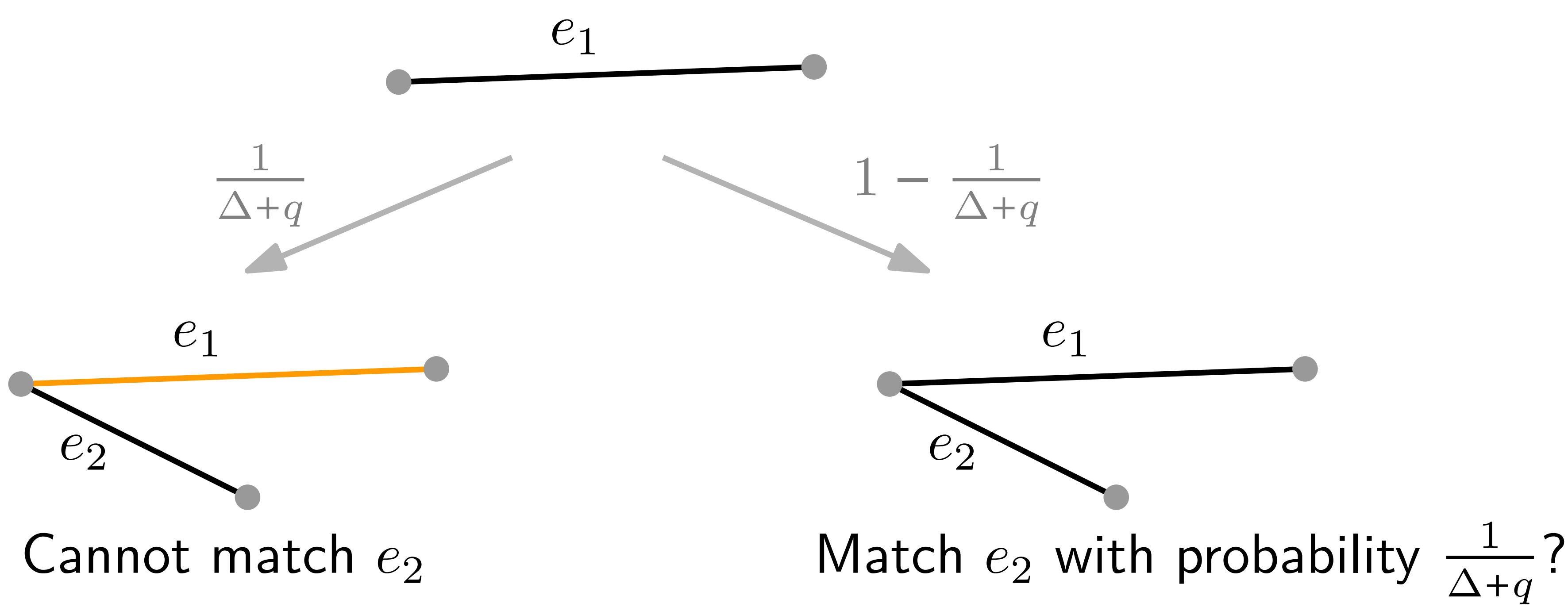
$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

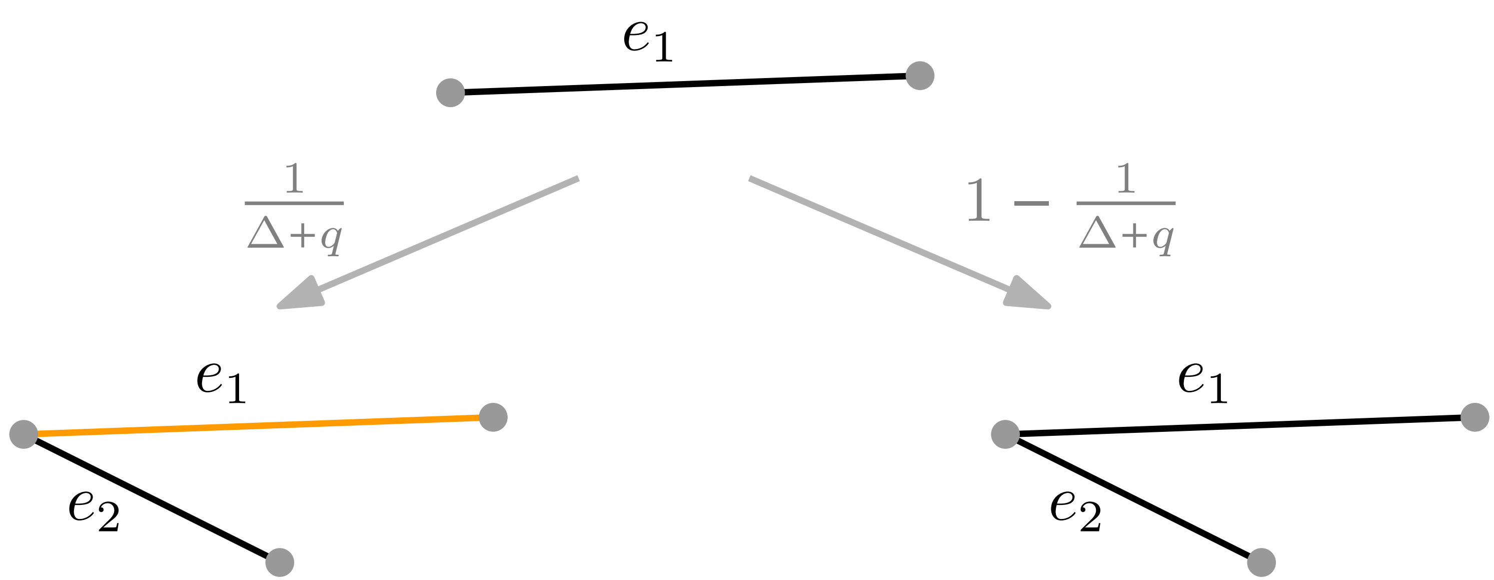
$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Cannot match e_2

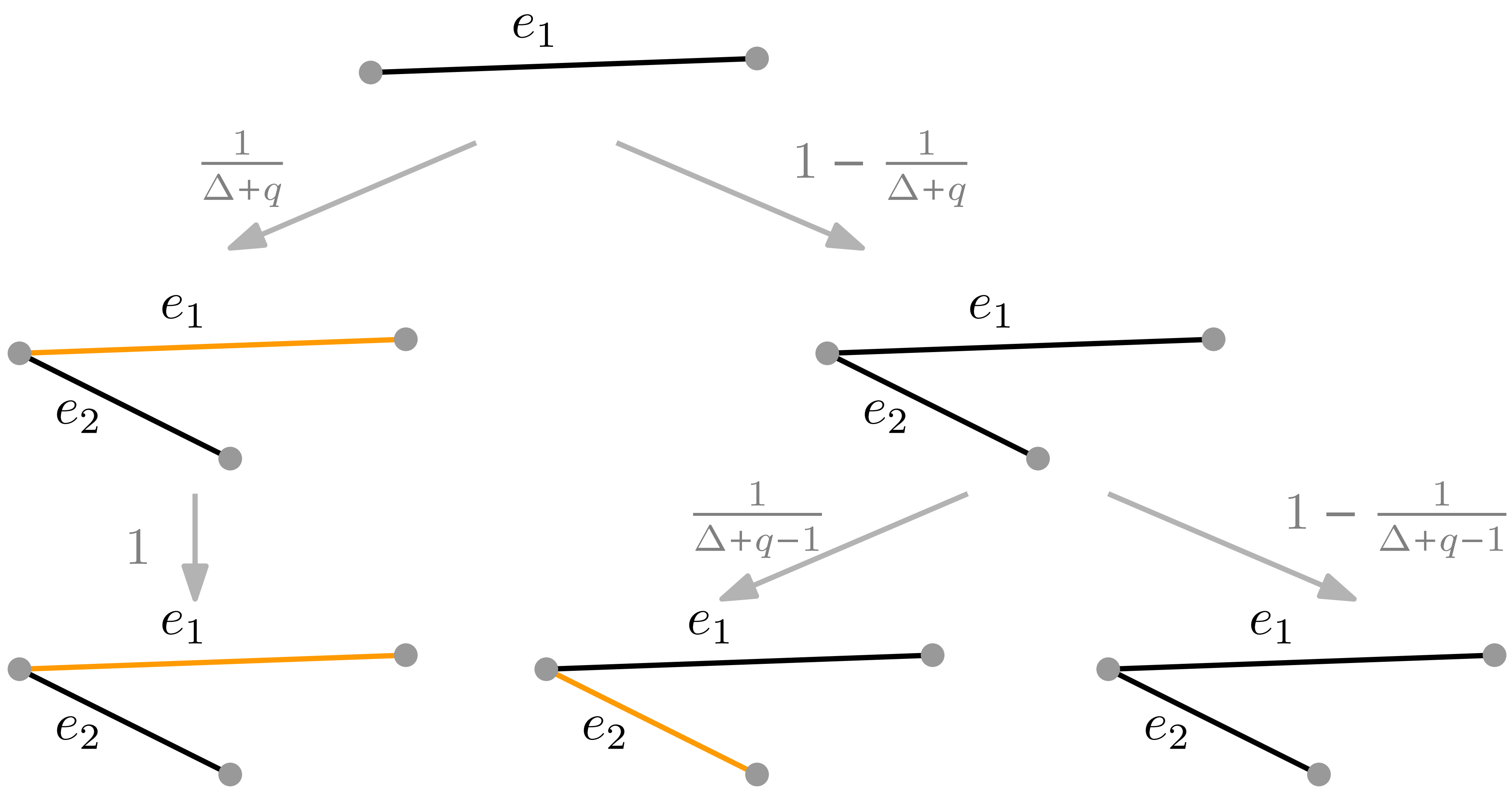
Match e_2 with probability $\frac{1}{\Delta+q}$?

Must scale up: $\frac{1}{\Delta+q} / (1 - \frac{1}{\Delta+q})$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta+q} \qquad q := \Theta(\Delta^{3/4}\sqrt{\log \Delta}) = o(\Delta)$$



Fair Matching — Natural First Attempt

When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

Fair Matching — Natural First Attempt

When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Potential Problem: $p_t > 1$

Fair Matching — Natural First Attempt

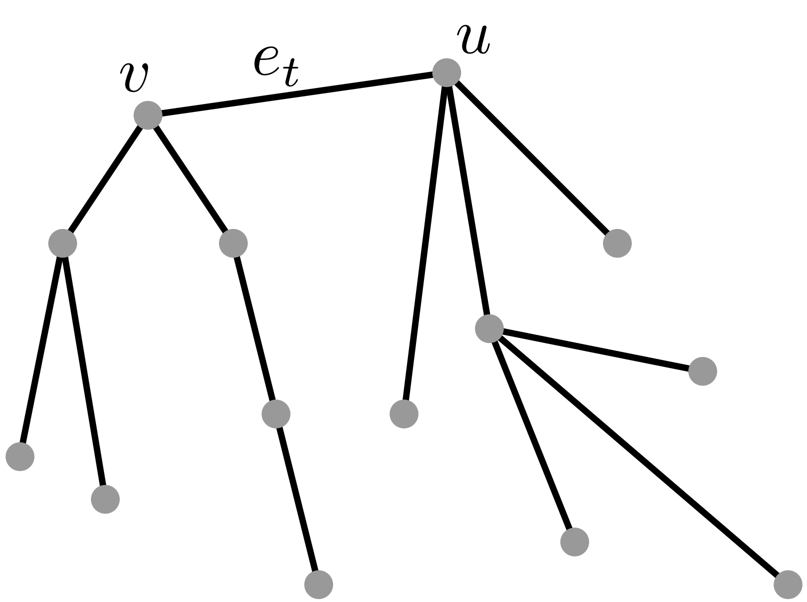
When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree

Potential Problem: $p_t > 1$



Fair Matching — Natural First Attempt

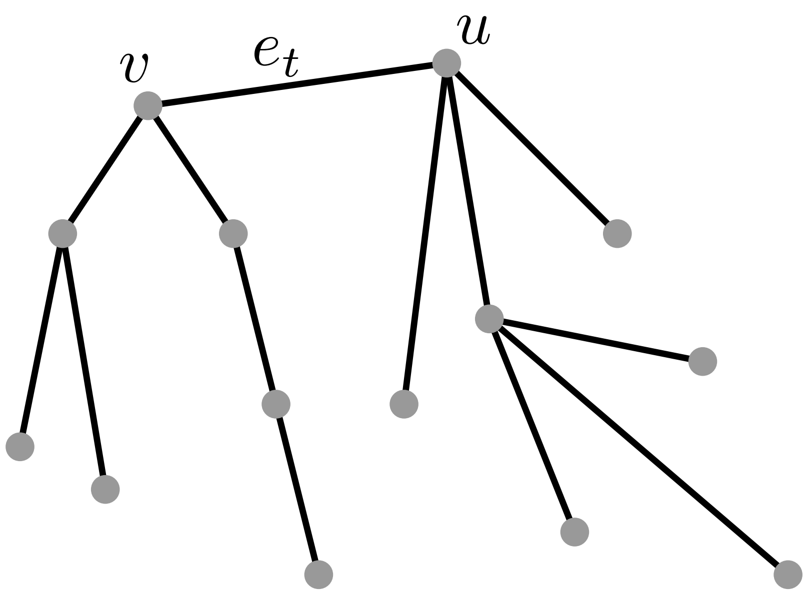
When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \dfrac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree

Potential Problem: $p_t > 1$



(tree)

$\Pr[\text{both } u, v \text{ free}] = \Pr[u \text{ free}] \cdot \Pr[v \text{ free}]$

Fair Matching — Natural First Attempt

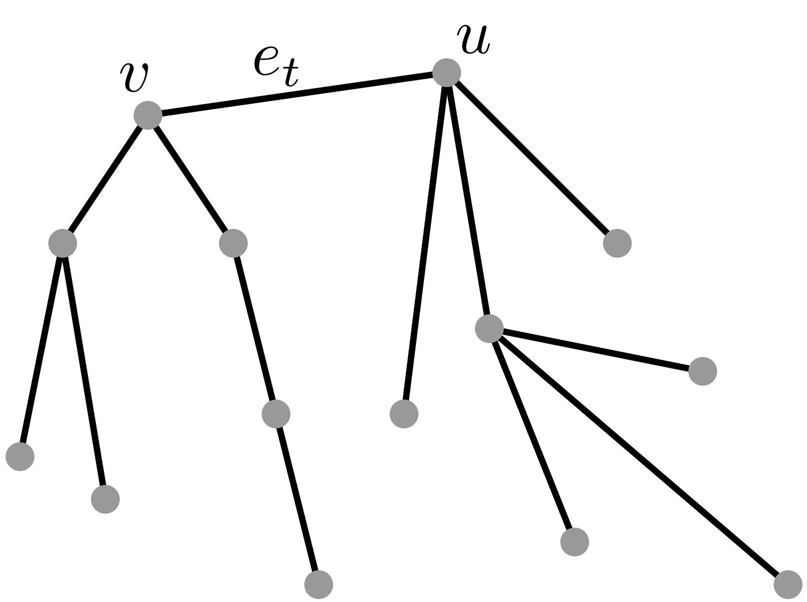
When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree

Potential Problem: $p_t > 1$



$$\Pr[\text{both } u, v \text{ free}] \overset{\text{(tree)}}{=} \Pr[u \text{ free}] \cdot \Pr[v \text{ free}]$$

$$\Pr[u \text{ free}] = 1 - \frac{\deg(u)}{\Delta+q} \geq \frac{q}{2\Delta}$$

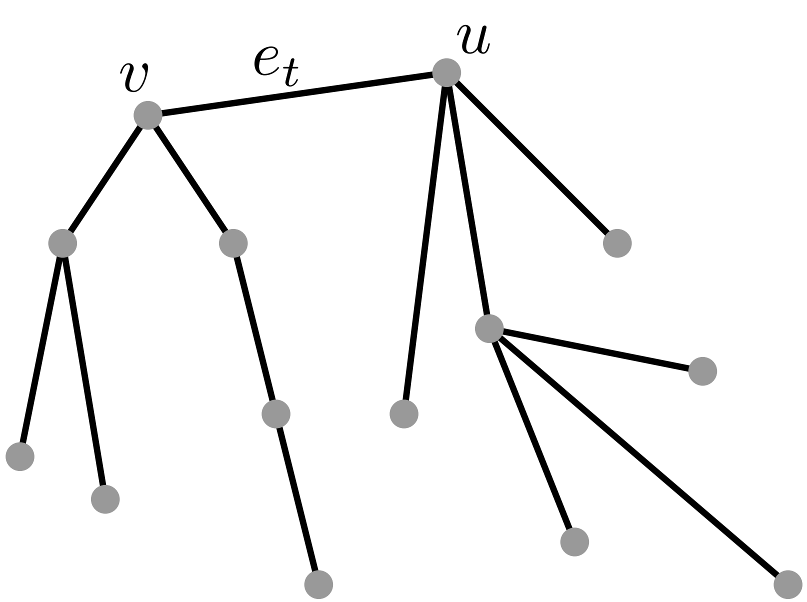
Fair Matching — Natural First Attempt

When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree



Potential Problem: $p_t > 1$

$$\Pr[\text{both } u, v \text{ free}] \overset{\text{(tree)}}{=} \Pr[u \text{ free}] \cdot \Pr[v \text{ free}]$$

$$\Pr[u \text{ free}] = 1 - \frac{\deg(u)}{\Delta+q} \geq \frac{q}{2\Delta}$$

$$p_t \leq \frac{1}{\Delta+q} \cdot \left(\frac{2\Delta}{q}\right)^2$$

Fair Matching — Natural First Attempt

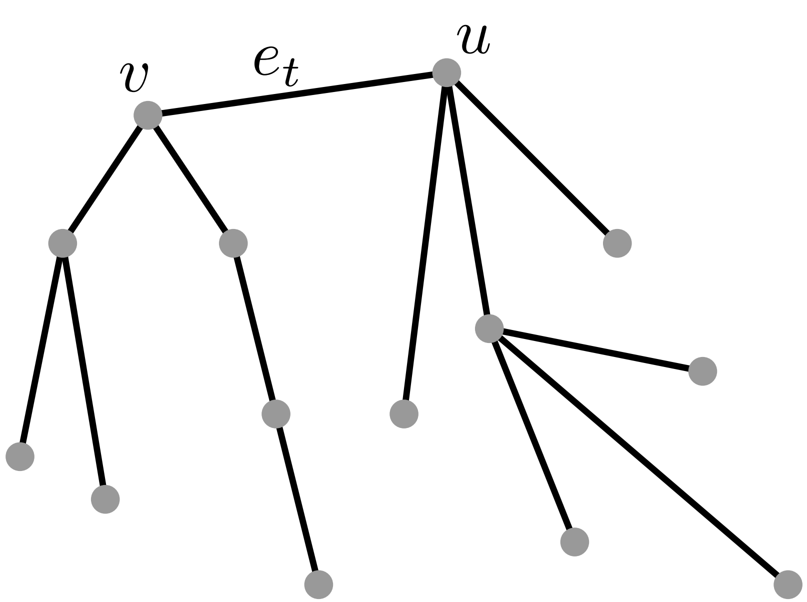
When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree

Potential Problem: $p_t > 1$



$$\Pr[\text{both } u, v \text{ free}] \overset{\text{(tree)}}{=} \Pr[u \text{ free}] \cdot \Pr[v \text{ free}]$$

$$\Pr[u \text{ free}] = 1 - \frac{\deg(u)}{\Delta+q} \geq \frac{q}{2\Delta}$$

$$p_t \leq \frac{1}{\Delta+q} \cdot \left(\frac{2\Delta}{q}\right)^2$$

$$\leq 1 \text{ if } q := 2\sqrt{\Delta}$$

Fair Matching — Natural First Attempt

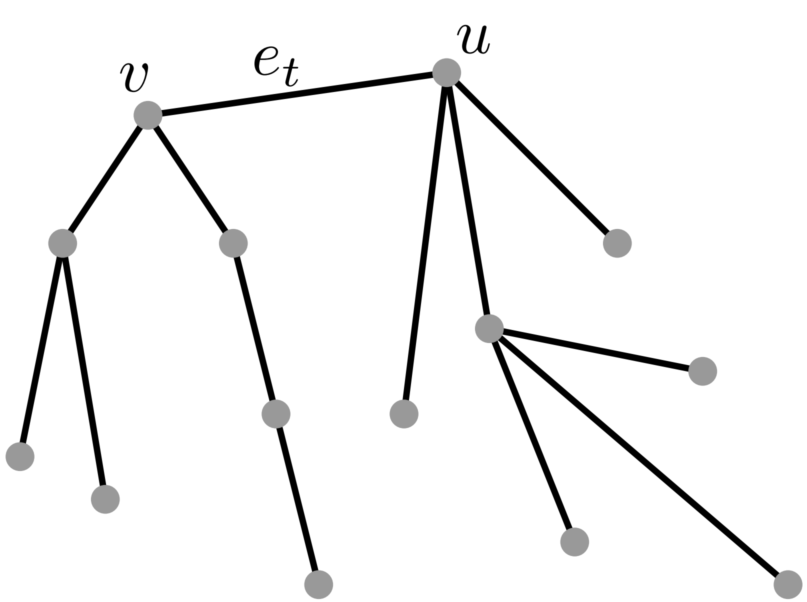
When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Example: G is a tree

Potential Problem: $p_t > 1$



$$\Pr[\text{both } u, v \text{ free}] \overset{\text{(tree)}}{=} \Pr[u \text{ free}] \cdot \Pr[v \text{ free}]$$

$$\Pr[u \text{ free}] = 1 - \frac{\deg(u)}{\Delta+q} \geq \frac{q}{2\Delta}$$

$$p_t \leq \frac{1}{\Delta+q} \cdot \left(\frac{2\Delta}{q}\right)^2$$

$$\leq 1 \text{ if } q := 2\sqrt{\Delta}$$

[Kulkarni/Liu/Sah/Sawhney/Tarnawski'22]

$(\frac{e}{e-1} + o(1))\Delta$-coloring subsampling locally tree-like graphs

Fair Matching — Natural First Attempt

When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Potential Problem: $p_t > 1$

Open Problem: Does this "Natural Algorithm" work in general?

Fair Matching — Natural First Attempt

When $e_t = (u, v)$ arrives:

Match e with probability $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives}]}$

$\implies \Pr[e \in M] = \frac{1}{\Delta+q}$.

Potential Problem: $p_t > 1$

Open Problem: Does this "Natural Algorithm" work in general?

Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free in } \textbf{current execution}]}$

Alternative Natural Algorithm

Our Alternative Algorithm: $p_t := \dfrac{1/(\Delta+q)}{\Pr[u, v \text{ both free in current execution}]}$

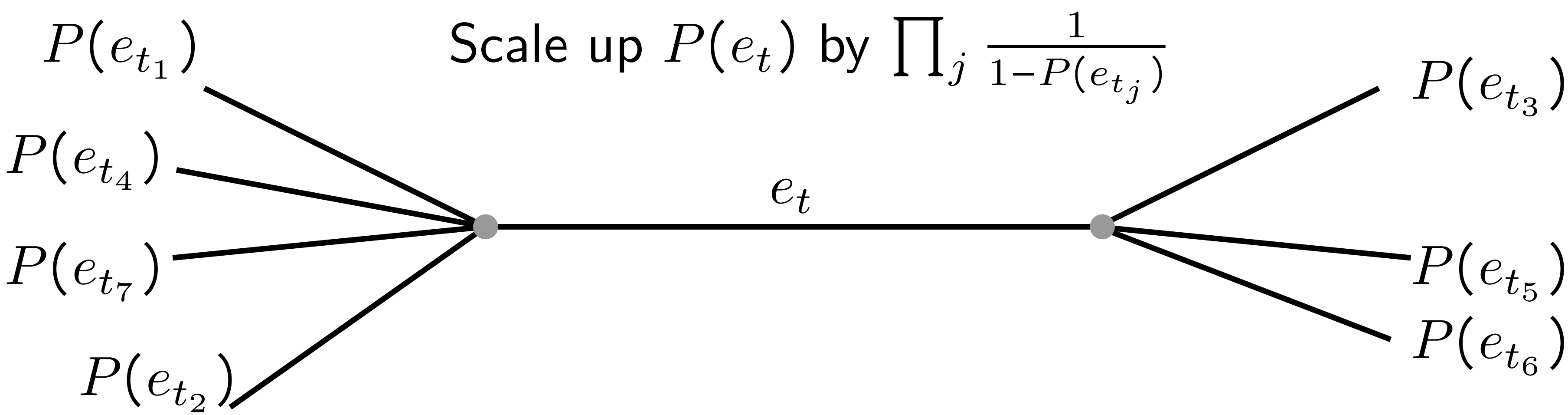
Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ *arrives, match it with probability*

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ *are those previously-arrived edges incident to the endpoints of* e_t.

Scale up $P(e_t)$ by $\prod_j \frac{1}{1-P(e_{t_j})}$



Alternative Natural Algorithm

Our Alternative Algorithm: $p_t := \dfrac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$

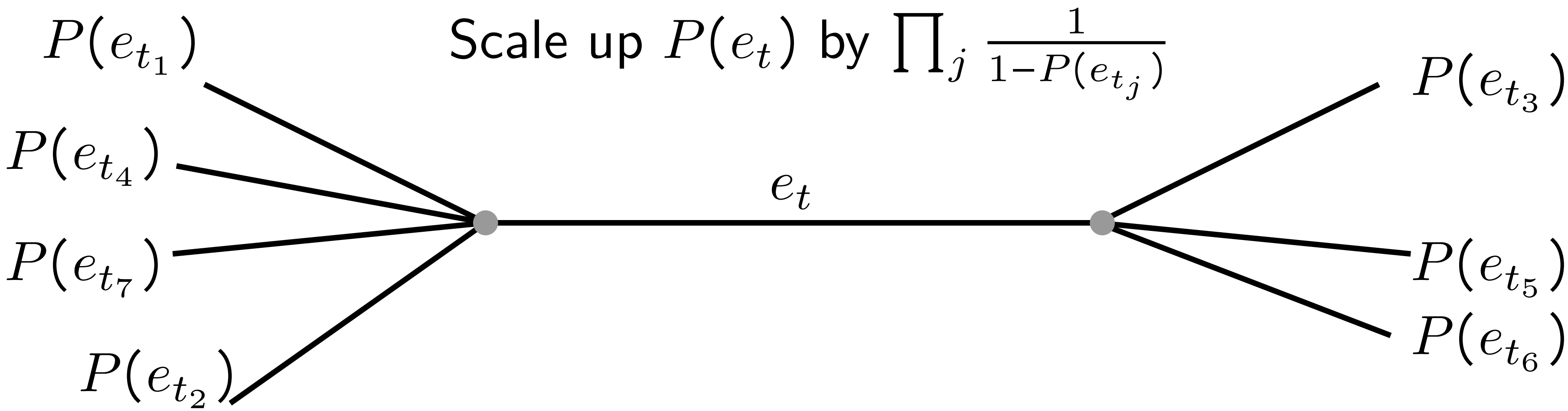
Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u,v)$ arrives, match it with probability

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Scale up $P(e_t)$ by $\prod_j \frac{1}{1-P(e_{t_j})}$



Alternative Natural Algorithm

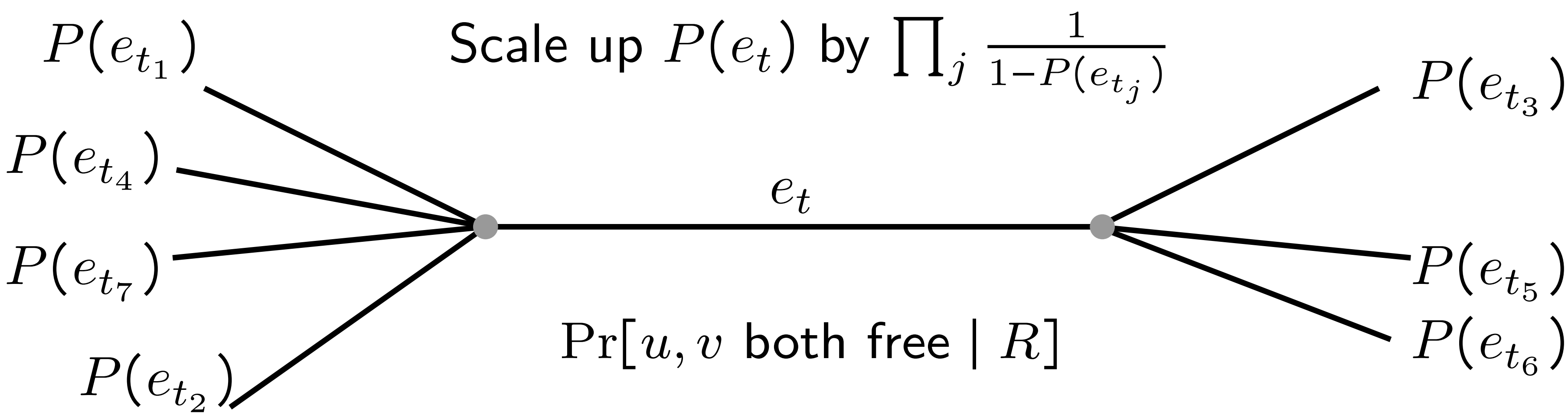
Our Alternative Algorithm: $p_t := \dfrac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$

Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.



R =randomness outside of incident edges.

Alternative Natural Algorithm

Algorithm 2 (MatchingAlgorithm).

Initialization: *Set* $F_1(v) \leftarrow 1$ *for every vertex* v *and* $M_1 \leftarrow \emptyset$.

At the arrival of edge $e_t = (u, v)$ at time t:

- *Sample* $X_t \sim Uni[0, 1]$.
- *Define*

$$P(e_t) = \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{F_t(u) \cdot F_t(v)} & \text{if } u \text{ and } v \text{ are unmatched in } M_t, \\ 0 & \text{otherwise.} \end{cases}$$

  and

$$\hat{P}(e_t) = \begin{cases} P(e_t) & \text{if } \min\{F_t(u), F_t(v)\} \cdot (1 - P(e_t)) \geqslant q/(4\Delta) \\ 0 & \text{otherwise.} \end{cases}$$

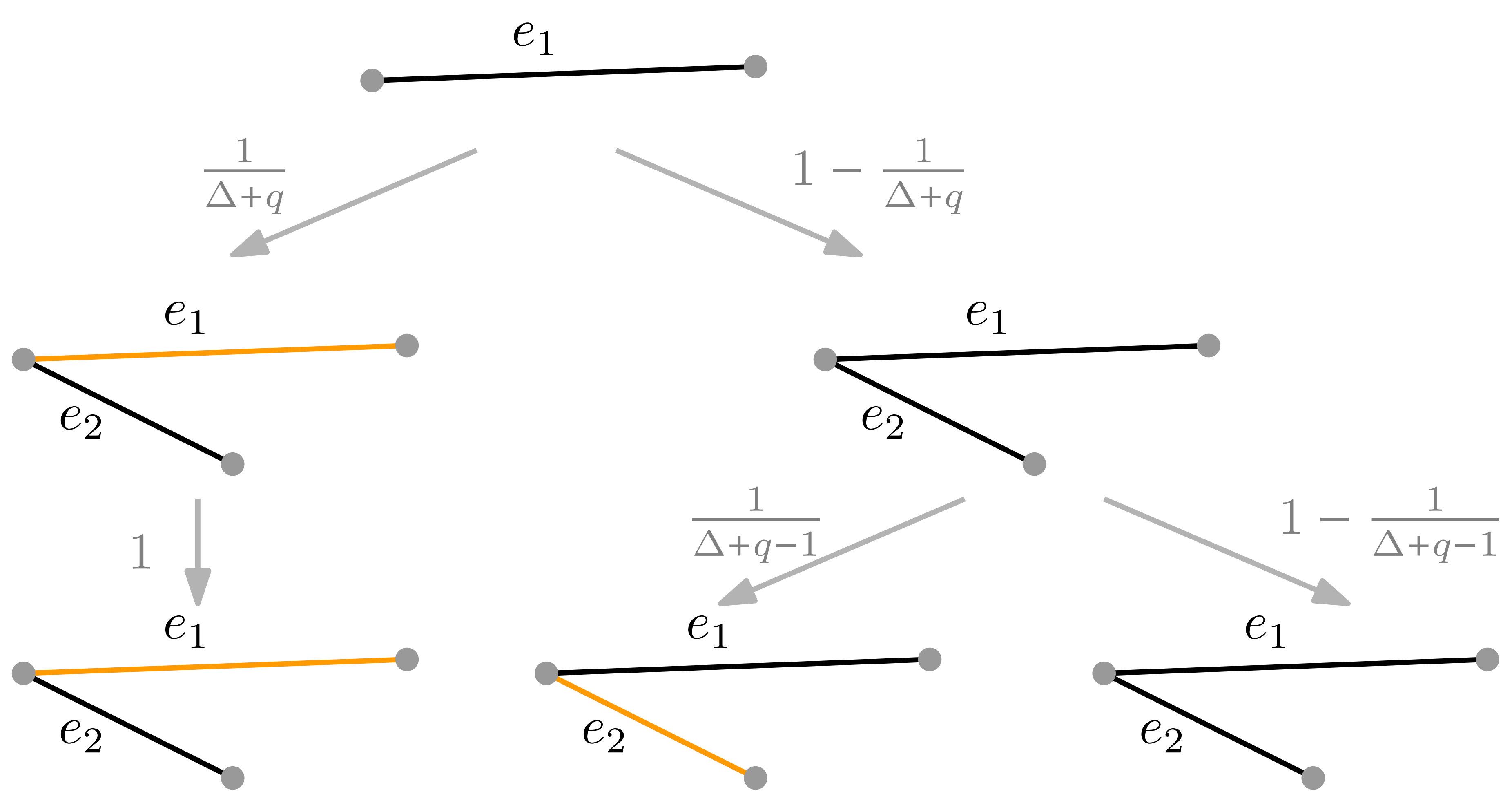
- *Set*
  - $F_{t+1}(u) \leftarrow F_t(u) \cdot (1 - \hat{P}(e_t))$;
  - $F_{t+1}(v) \leftarrow F_t(v) \cdot (1 - \hat{P}(e_t))$;
  - $M_{t+1} \leftarrow \begin{cases} M_t \cup \{e_t\} & \text{if } X_t < \hat{P}(e_t), \\ M_t & \text{otherwise.} \end{cases}$

A More Fine-Grained Bayesian Approach

Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$

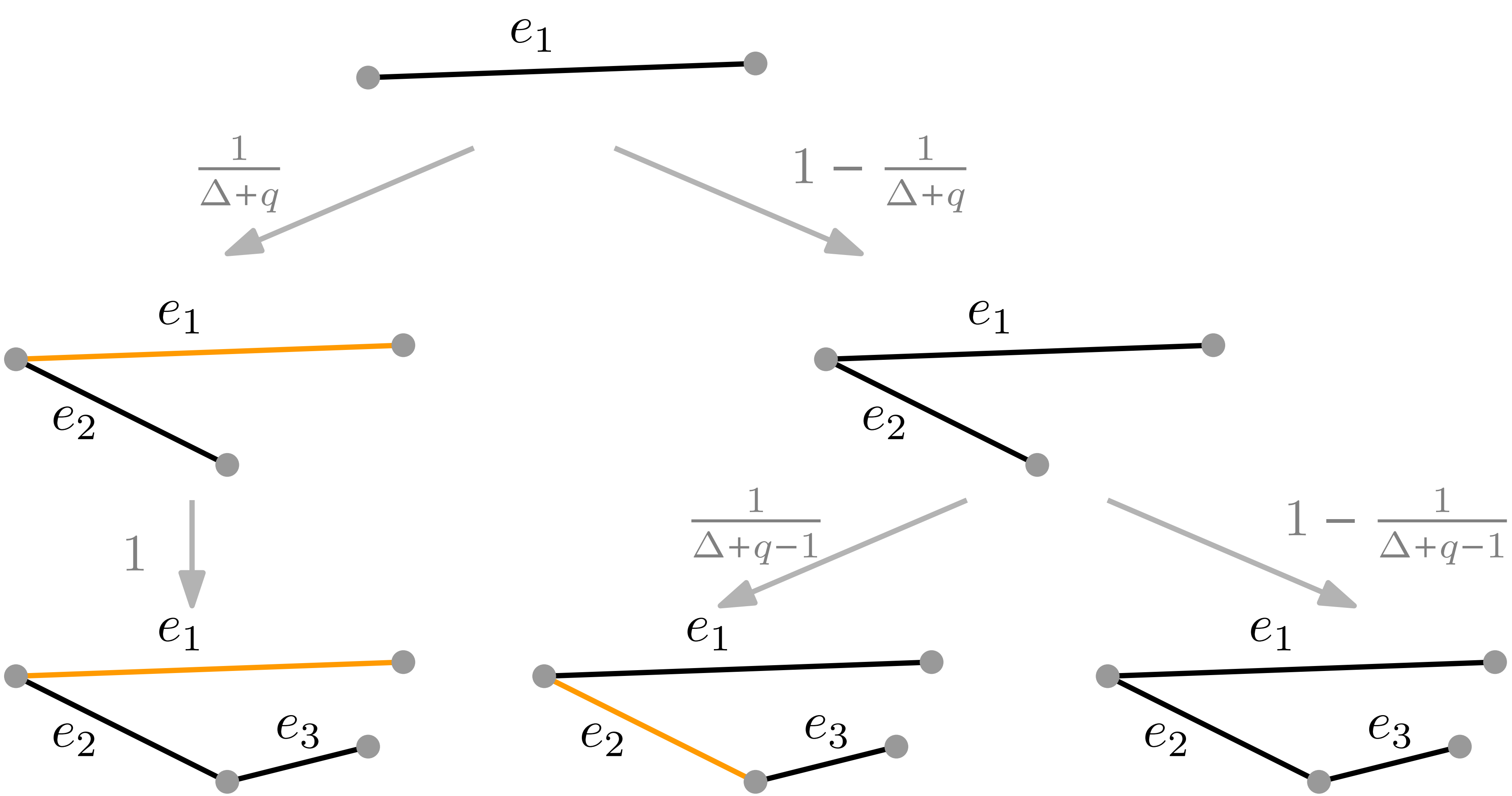
A More Fine-Grained Bayesian Approach

Our Alternative Algorithm: $p_t := \dfrac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$



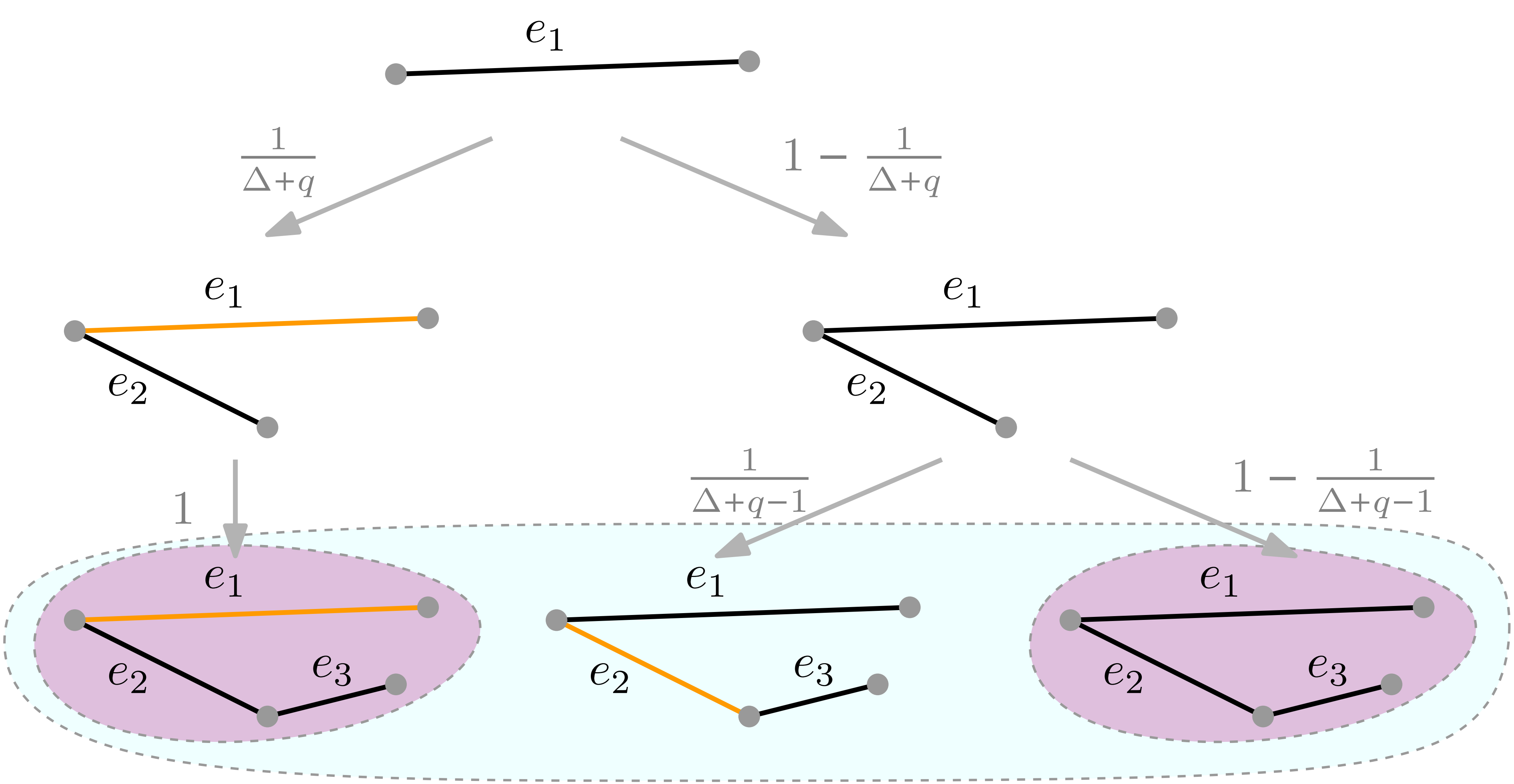
A More Fine-Grained Bayesian Approach

Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$



A More Fine-Grained Bayesian Approach

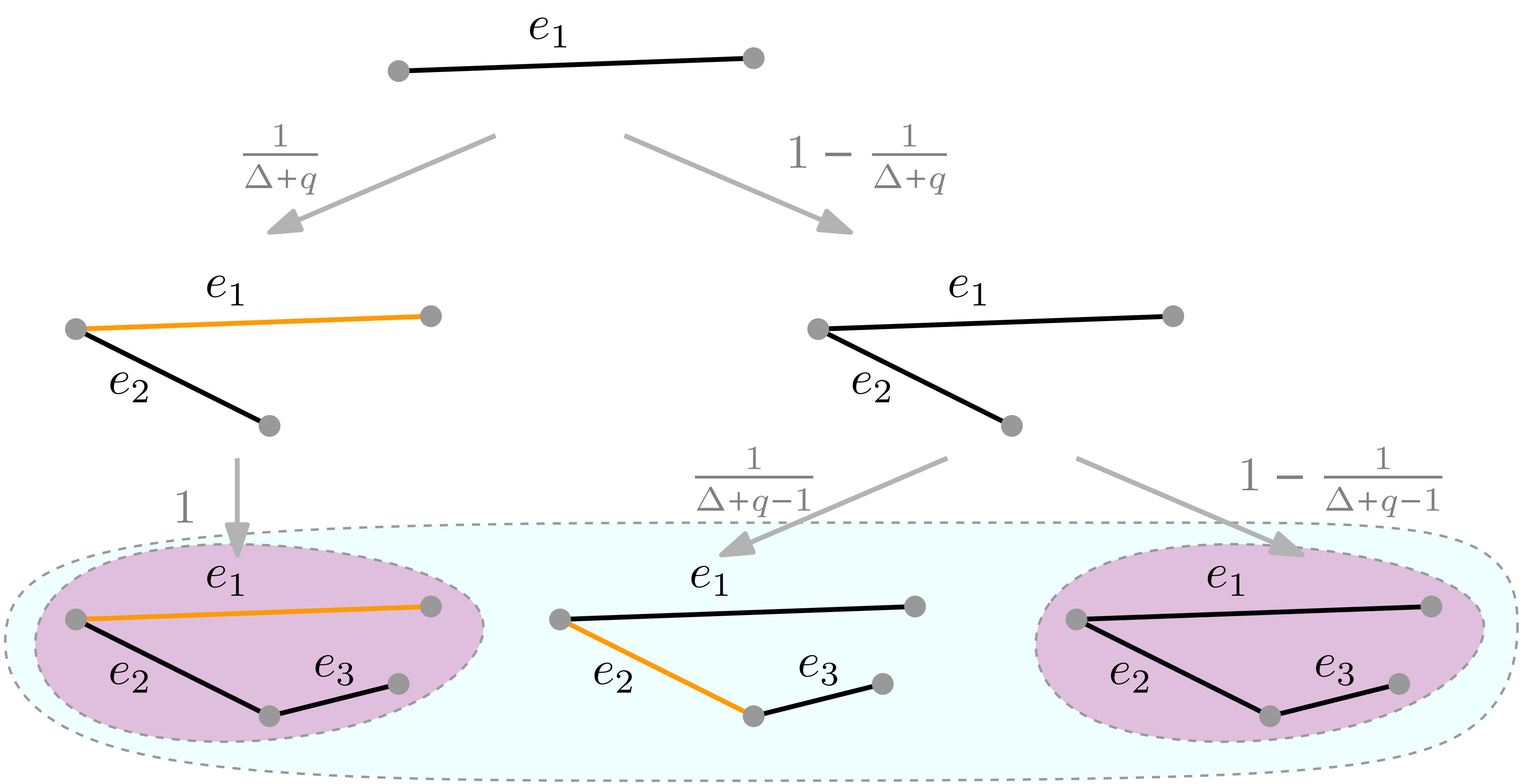
Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$



Old Algo: $\Pr[u,v \text{ free}] = (1 - \frac{1}{\Delta+q})$ Uses same scaling factor

A More Fine-Grained Bayesian Approach

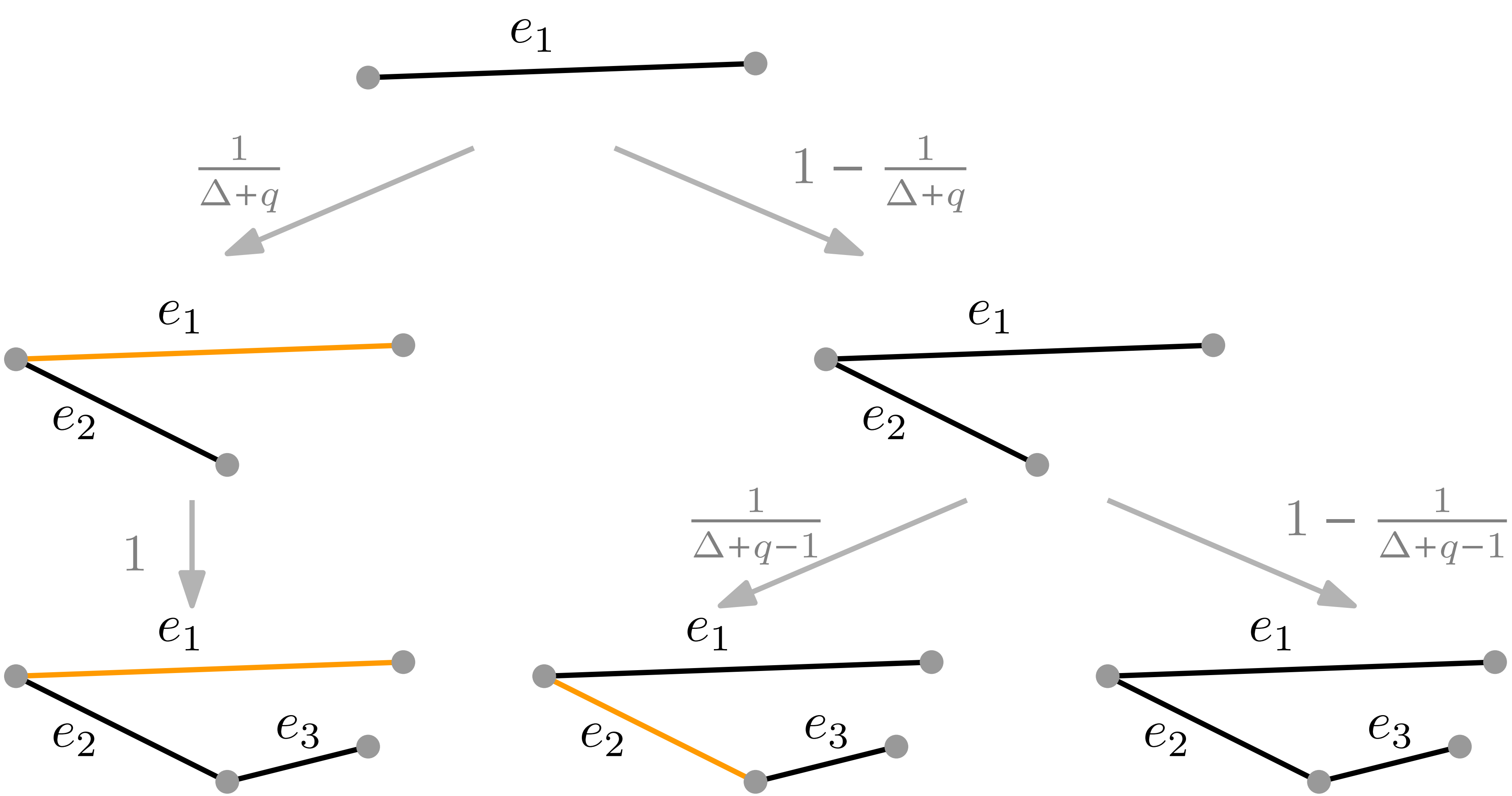
Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$



Old Algo: $\Pr[u,v \text{ free}] = (1 - \frac{1}{\Delta+q})$ Uses same scaling factor

A More Fine-Grained Bayesian Approach

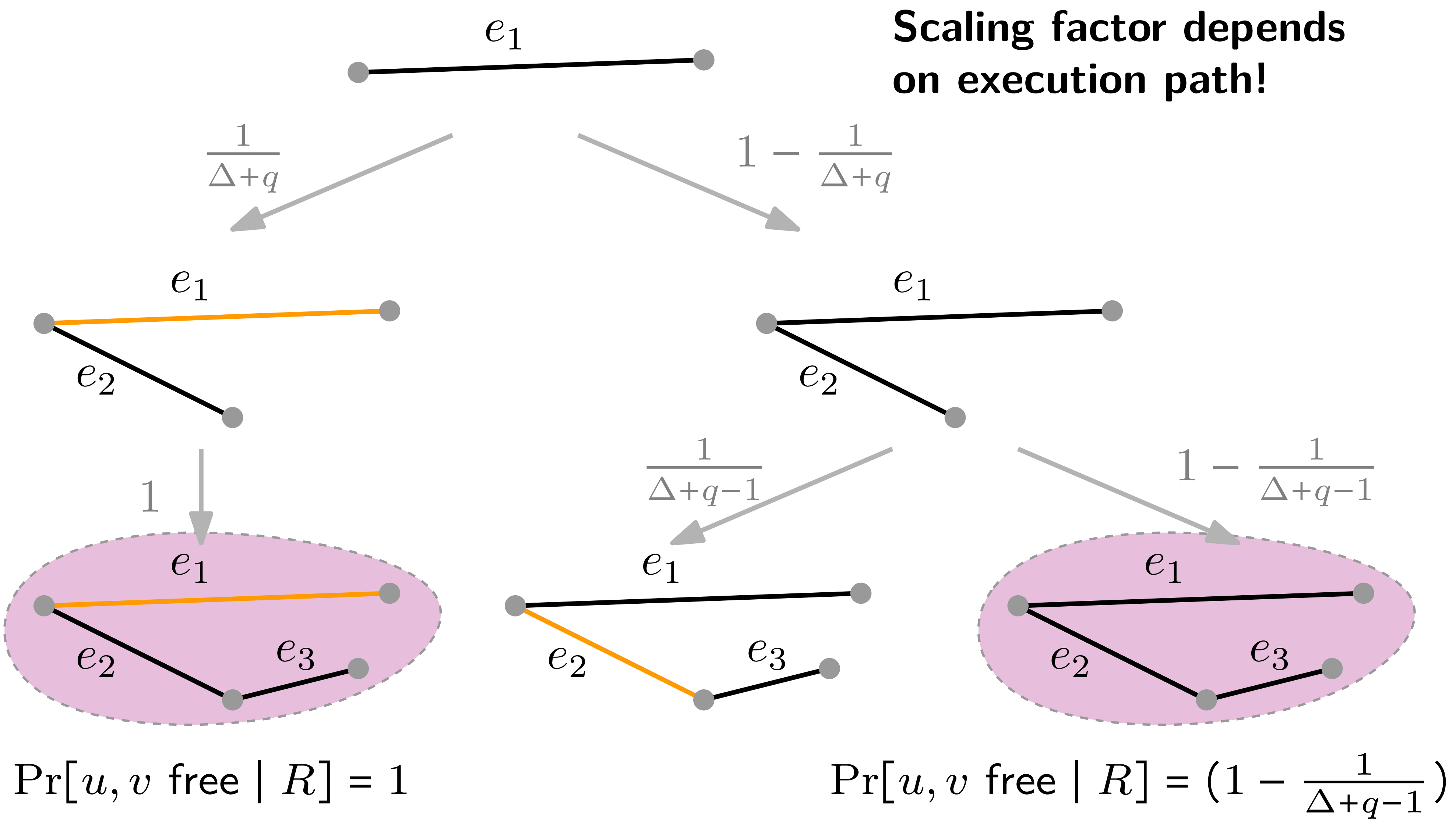
Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u, v \text{ both free in current execution}]}$



A More Fine-Grained Bayesian Approach

Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$

Scaling factor depends on execution path!

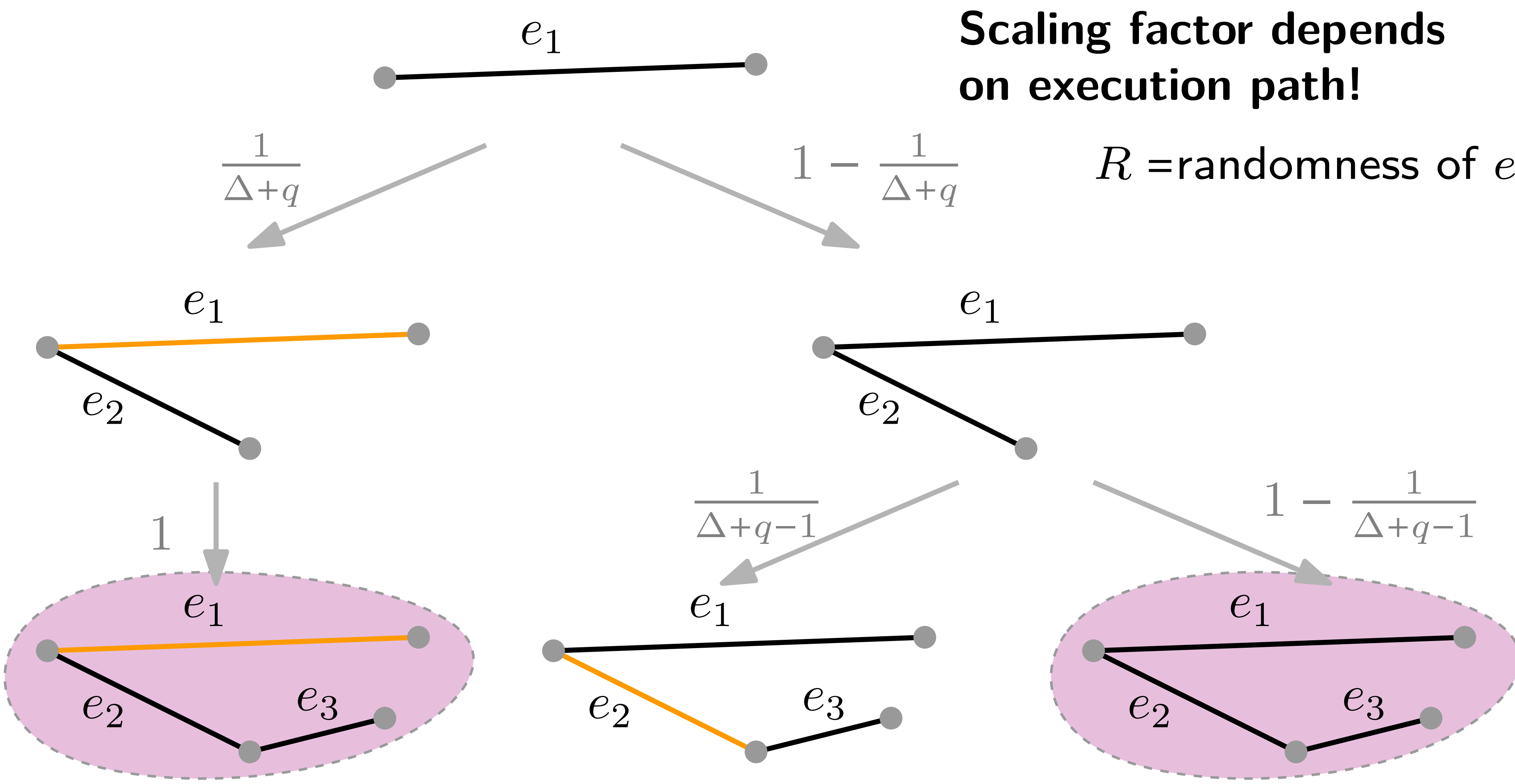


A More Fine-Grained Bayesian Approach

Our Alternative Algorithm: $p_t := \frac{1/(\Delta+q)}{\Pr[u,v \text{ both free in current execution}]}$

Scaling factor depends on execution path!

R =randomness of e_1



$\Pr[u,v \text{ free} \mid R] = 1$

$\Pr[u,v \text{ free} \mid R] = (1-\frac{1}{\Delta+q-1})$

Analysis Idea — Random Walk

Core of Analysis: Prove $P(e_t) \leq 1$

Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Core of Analysis: Prove $P(e_t) \leq \frac{10}{\sqrt{\Delta}}$

Previous work: Control Correlation
Our work: Embrace Correlations

Algorithm 1 (NaturalMatchingAlgorithm).

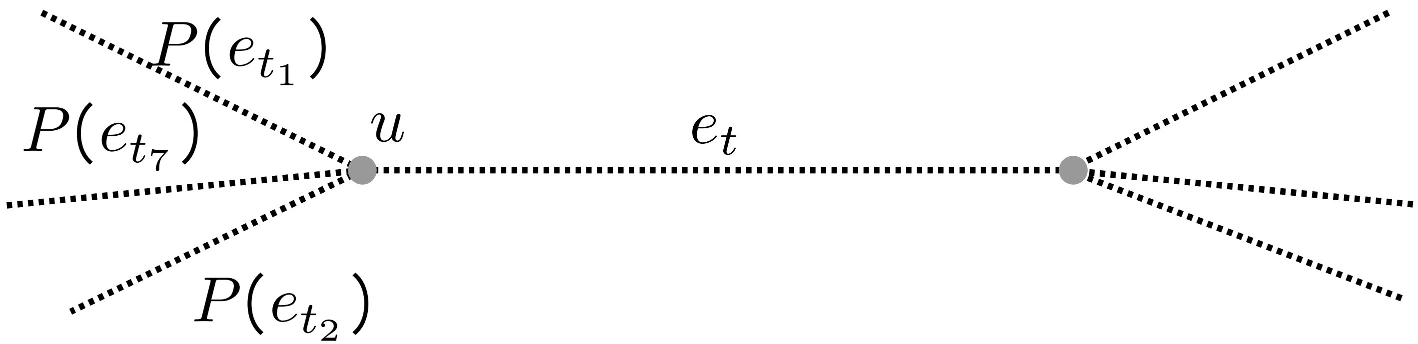
When an edge $e_t = (u, v)$ arrives, match it with probability

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

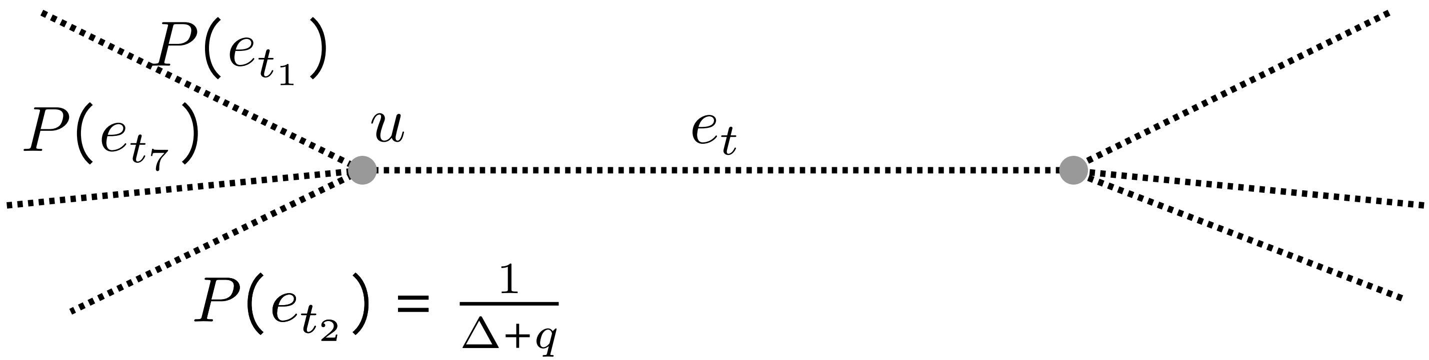
$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$

Goal: Show $S_u \gtrsim \sqrt[4]{\frac{1}{10\Delta}}$



Algorithm 1 (NaturalMatchingAlgorithm).

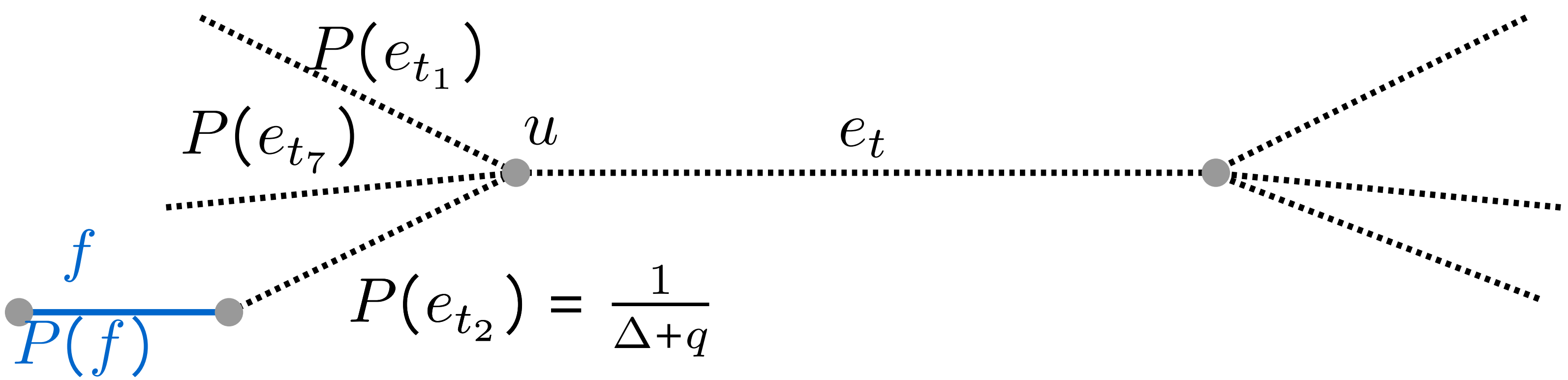
When an edge $e_t = (u, v)$ arrives, match it with probability

$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$ **Goal:** Show $S_u \gtrsim \sqrt[4]{\frac{1}{10\Delta}}$



Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

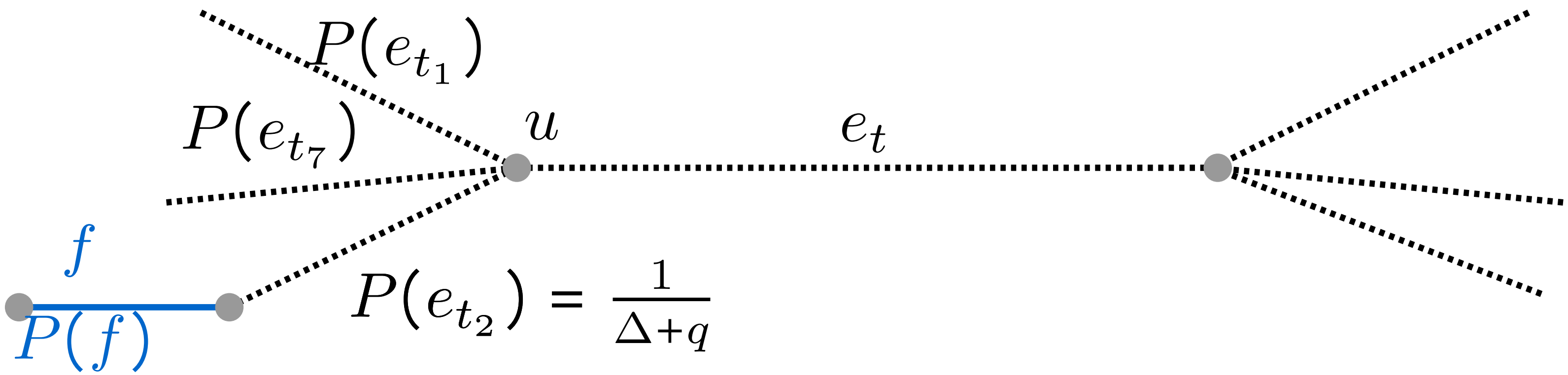
$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$

Goal: Show $S_u \gtrsim \sqrt[4]{\frac{1}{10\Delta}}$



If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$

If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

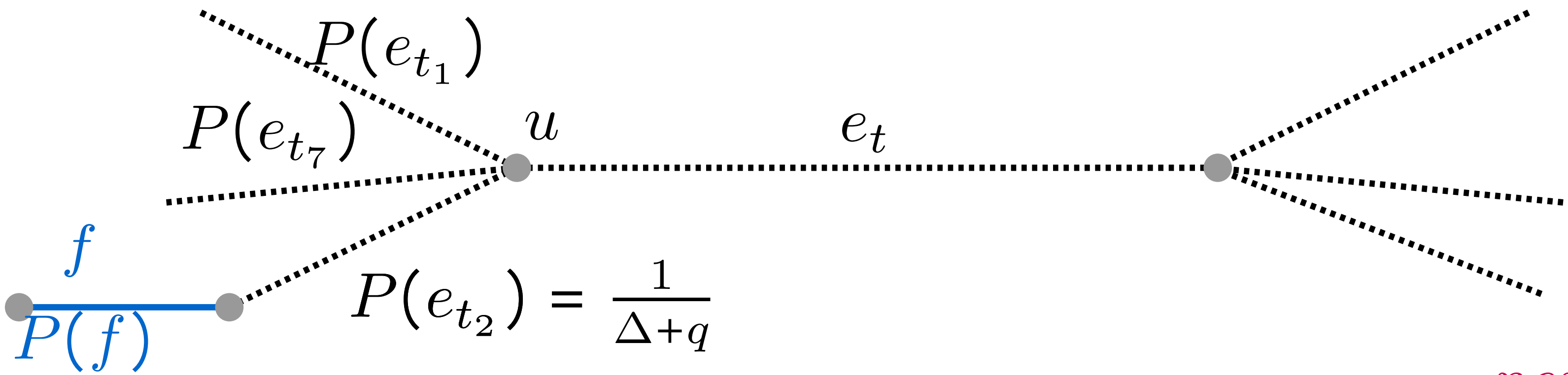
$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \text{if } u \text{ and } v \text{ are still unmatched,} \\ 0 & \text{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$

Goal: Show $S_u \gtrsim \sqrt[4]{\frac{1}{10\Delta}}$



If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$

If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

$\mathbb{E}[S_u^{new}] = S_u$

Algorithm 1 (NaturalMatchingAlgorithm).

When an edge $e_t = (u, v)$ arrives, match it with probability

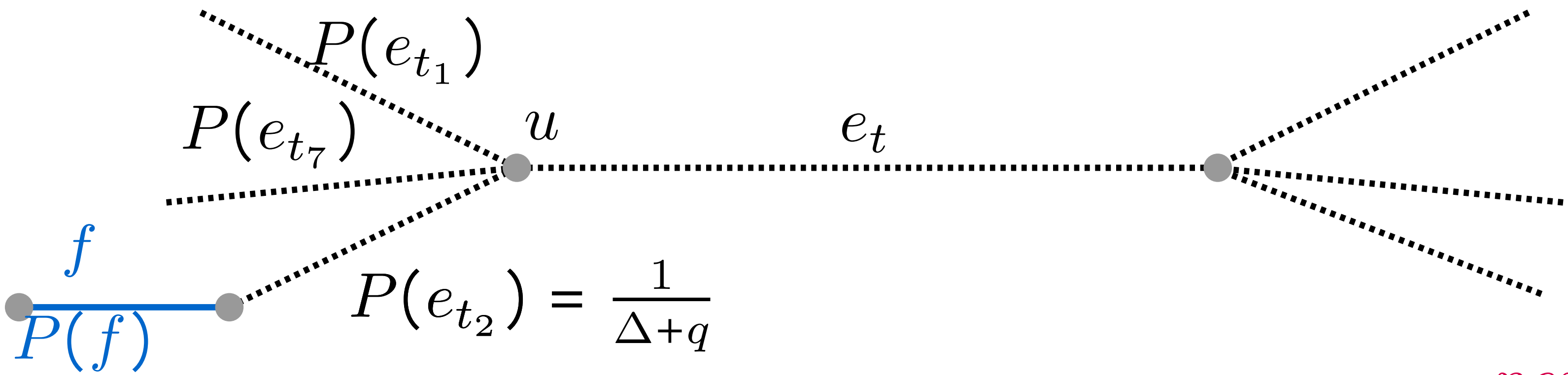
$$P(e_t) \leftarrow \begin{cases} \frac{1}{\Delta+q} \cdot \frac{1}{\prod_{j=1}^{k}(1-P(e_{t_j}))} & \textit{if } u \textit{ and } v \textit{ are still unmatched,} \\ 0 & \textit{otherwise,} \end{cases}$$

where $e_{t_1}, \ldots, e_{t_k}$ are those previously-arrived edges incident to the endpoints of e_t.

Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$

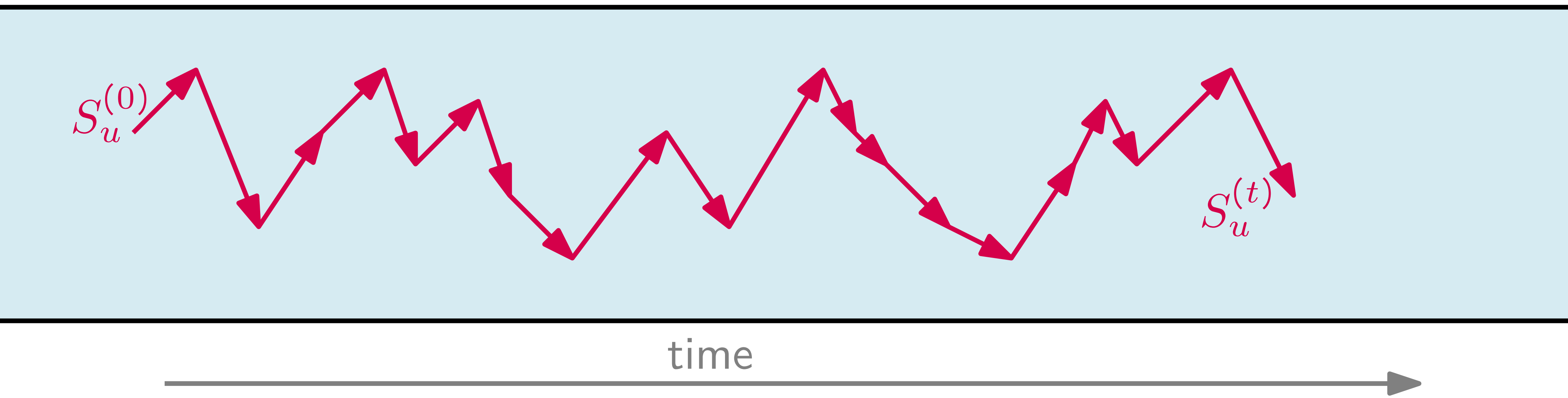
Goal: Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$

If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

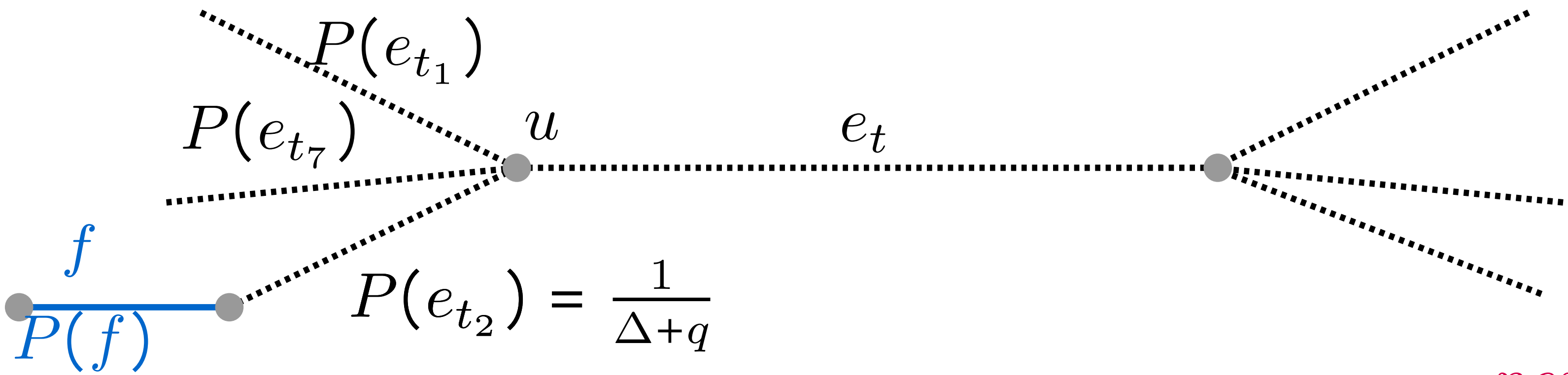
$\mathbb{E}[S_u^{new}] = S_u$



Analysis Idea — Random Walk

Scaling factor $S_u := (1 - \sum_j P(e_{t_j}))$

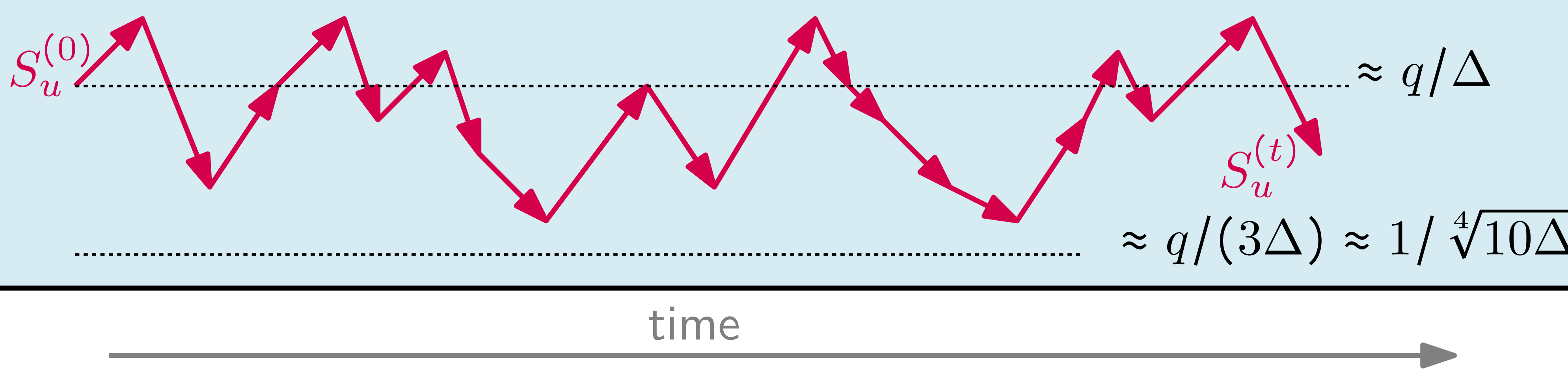
Goal: Show $S_u \gtrsim \sqrt[4]{\frac{1}{10\Delta}}$



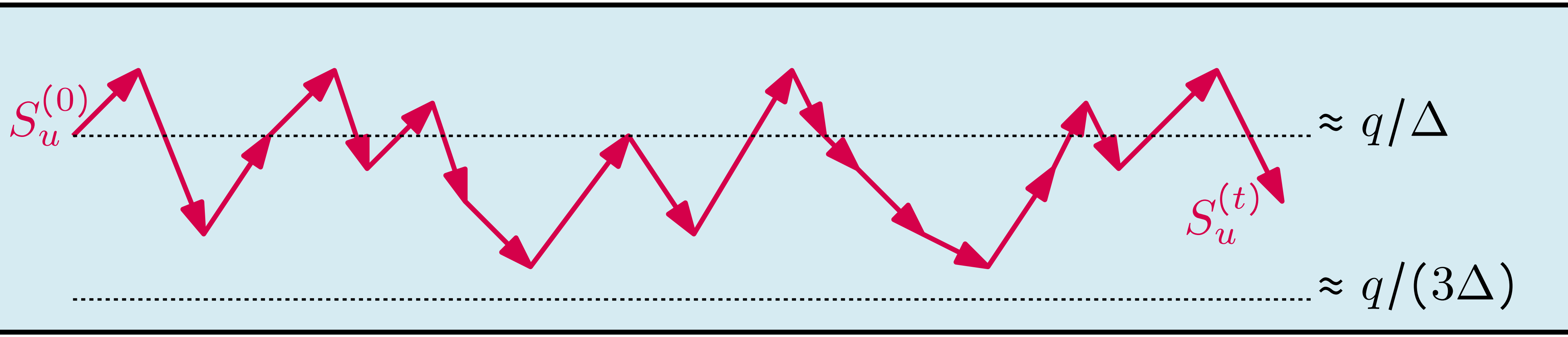
If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$

If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

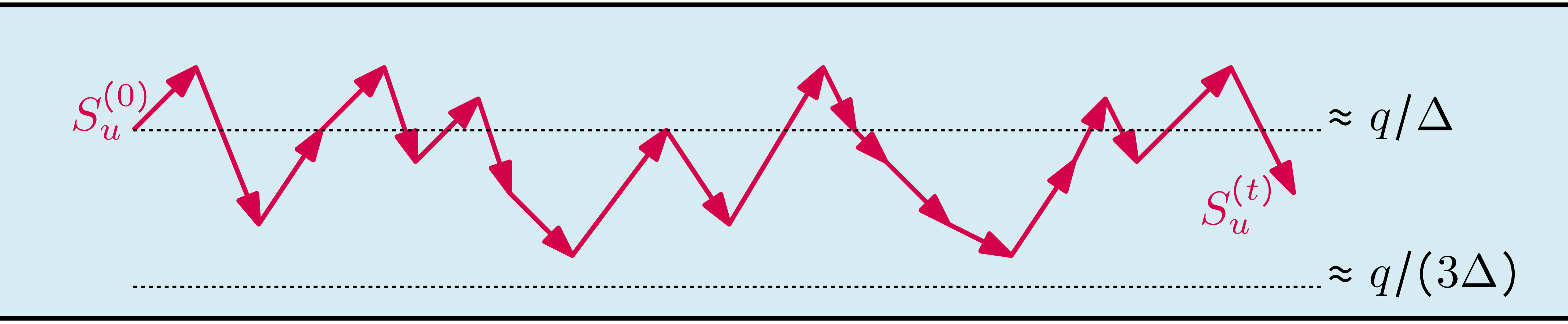
$\mathbb{E}[S_u^{new}] = S_u$



Martingale Process

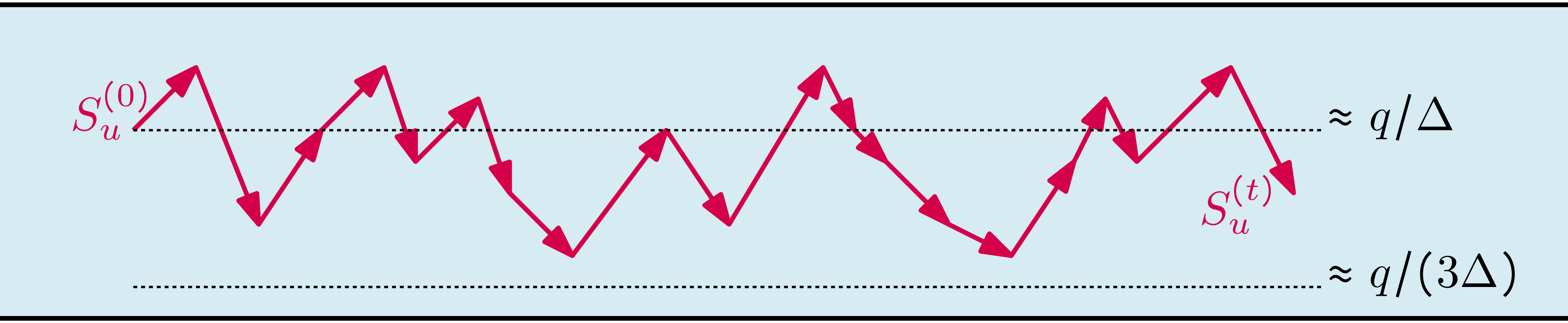


Martingale Process



(If no correlations: Chernoff bound) $q :\approx \Delta^{3/4}$

Martingale Process



(If no correlations: Chernoff bound) $q :\approx \Delta^{3/4}$

Freedmans Inequality:

- Martingale $\mathbb{E}[Z_{t+1} - Z_t \mid Z_1, Z_2, \ldots, Z_t] = 0$
- Step size $|Z_{t+1} - Z_t| \leq A$
- Observed variance $\sum_t \mathbb{E}\left[(Z_{t+1} - Z_t)^2 \mid Z_1, \ldots, Z_t\right] \leq \sigma^2$

Martingale Process

(If no correlations: Chernoff bound) $q :\approx \Delta^{3/4}$

Freedmans Inequality:

- Martingale $\mathbb{E}[Z_{t+1} - Z_t \mid Z_1, Z_2, \ldots, Z_t] = 0$
- Step size $|Z_{t+1} - Z_t| \leq A$
- Observed variance $\sum_t \mathbb{E}\left[(Z_{t+1} - Z_t)^2 \mid Z_1, \ldots, Z_t\right] \leq \sigma^2$

$$\implies \Pr\left[|Z_t - Z_0| \geq \varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2}{2(\sigma^2 + A\varepsilon/3)}\right)$$

Fair Matching Result

Main Technical Result:
There is an online algorithm which outputs a random matching M so that

$$\Pr[e \in M] \geq \frac{1}{\Delta + q} \quad \forall e \in E, \quad \text{where} \quad q = O(\Delta^{3/4}\sqrt{\log \Delta})$$

Summary and Open Problems

Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.

Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) "as easy as offline":

Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) "as easy as offline":

Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) “as easy as offline”:

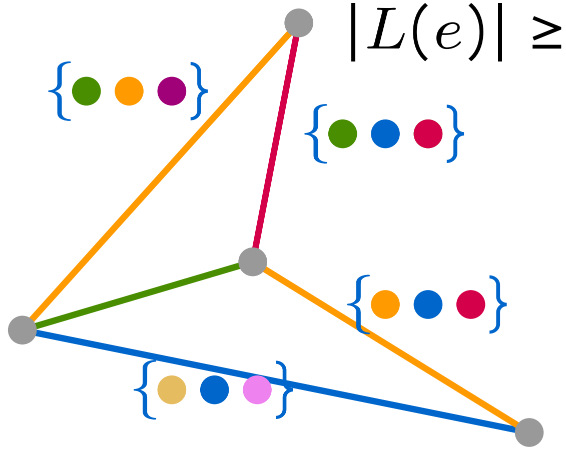
Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

List edge coloring

lists $L(e)$ of allowed colors

$|L(e)| \geq (1 + o(1))\Delta$



Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) "as easy as offline":

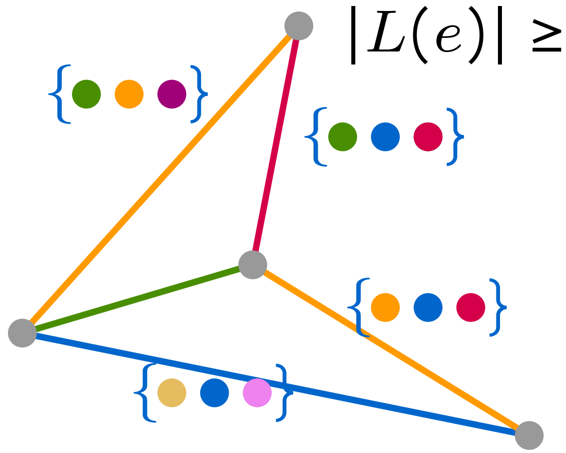
Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

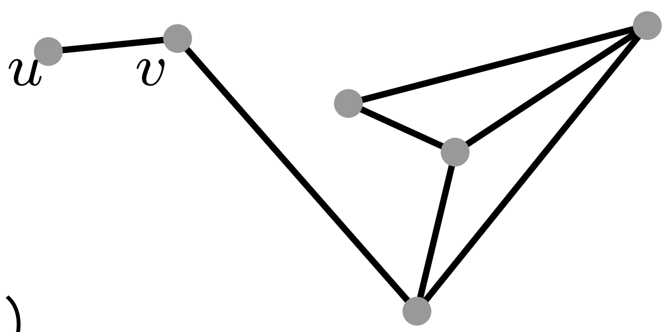
List edge coloring

lists $L(e)$ of allowed colors

$|L(e)| \geq (1 + o(1))\Delta$



Local edge coloring



color(u, v)
$\leq (1 + o(1)) \max(\deg(u), \deg(v))$
$+O(\log n)$

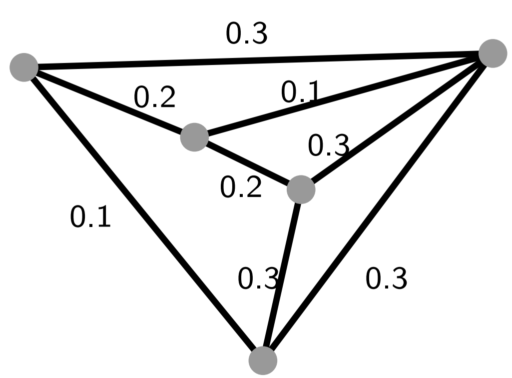
Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) "as easy as offline":

Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

Online Rounding of "Spread Out" Fractional Matchings:
Given (online) fractional matching $x \in \mathbb{R}^E$ satisfying $x_e \leq \varepsilon^5$,
output matching M so that $\Pr[e \in M] \geq (1 - \varepsilon)x_e$.



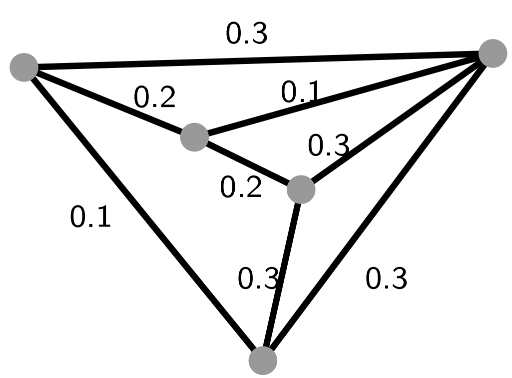
Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) "as easy as offline":

Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

Online Rounding of "Spread Out" Fractional Matchings:
Given (online) fractional matching $x \in \mathbb{R}^E$ satisfying $x_e \leq \varepsilon^5$,
output matching M so that $\Pr[e \in M] \geq (1 - \varepsilon)x_e$.



Also works in non-bipartite graphs depsite the integrality gap

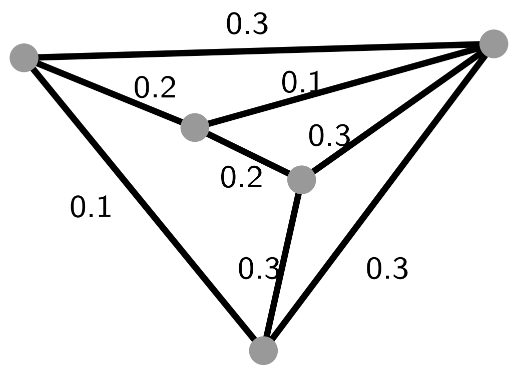
Summary

- For low-deg graphs $(2\Delta - 1)$-edge-coloring is optimal.
- Otherwise, edge coloring is (nearly) “as easy as offline”:

Main Theorem: online $(1 + o(1))$-fair matching algorithm
Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

- Extensions / Generalizations:

Online Rounding of “Spread Out” Fractional Matchings:
Given (online) fractional matching $x \in \mathbb{R}^E$ satisfying $x_e \leq \varepsilon^5$,
output matching M so that $\Pr[e \in M] \geq (1 - \varepsilon)x_e$.



Also works in non-bipartite graphs
depsite the integrality gap

$x_e := \frac{1}{\Delta}$ recovers fair matching theorem

Open Problems

- Deterministic?
  - (Or equiv. randomized vs adaptive aversary)
  - Completely open if one can beat greedy

Open Problems

- Deterministic?
  - (Or equiv. randomized vs adaptive aversary)
  - Completely open if one can beat greedy
- Correct Asymptotics:
  - Extra colors needed between
    $$\Omega(\sqrt{\Delta} + \log n) \quad \text{and} \quad O(\Delta^{3/4}\sqrt{\log \Delta} + \Delta^{2/3}\log^{1/3} n)$$
  - Beat greedy, or improve LB for $\Delta \in [\sqrt{\log n}, \log n]$?

Open Problems

- Deterministic?
  - (Or equiv. randomized vs adaptive aversary)
  - Completely open if one can beat greedy
- Correct Asymptotics:
  - Extra colors needed between

    $$\Omega(\sqrt{\Delta} + \log n) \quad \text{and} \quad O(\Delta^{3/4}\sqrt{\log \Delta} + \Delta^{2/3}\log^{1/3} n)$$
  - Beat greedy, or improve LB for $\Delta \in [\sqrt{\log n}, \log n]$?
- Multigraphs?
  - Offline: $\min(\frac{3}{2}\Delta, \Delta + \mu)$ colors
  - Or even hypergraphs?

Open Problems

- Deterministic?
  - (Or equiv. randomized vs adaptive aversary)
  - Completely open if one can beat greedy
- Correct Asymptotics:
  - Extra colors needed between
    $$\Omega(\sqrt{\Delta} + \log n) \quad \text{and} \quad O(\Delta^{3/4}\sqrt{\log \Delta} + \Delta^{2/3}\log^{1/3} n)$$
  - Beat greedy, or improve LB for $\Delta \in [\sqrt{\log n}, \log n]$?
- Multigraphs?
  - Offline: $\min(\frac{3}{2}\Delta, \Delta + \mu)$ colors
  - Or even hypergraphs?

Thanks!