Approximating Shortest Connected Graph Transformation for Trees

SOFSEM 2020

Nicolas Bousquet, Alice Joffard

January 21, 2020

Shortest Graph Transformation

Shortest Graph Transformation

SHORTEST GRAPH TRANSFORMATION (SGT)

Shortest Graph Transformation

SHORTEST GRAPH TRANSFORMATION (SGT)

INPUT: Two multigraphs G,H with the same vertices and the same degree sequence, an integer k

Shortest Graph Transformation

SHORTEST GRAPH TRANSFORMATION (SGT)

INPUT: Two multigraphs G,H with the same vertices and the same degree sequence, an integer k

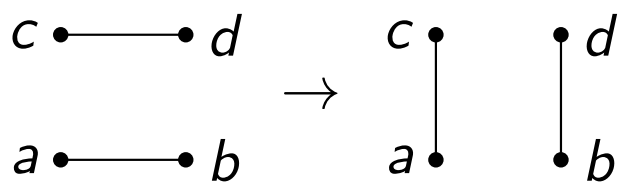
OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H

Shortest Graph Transformation

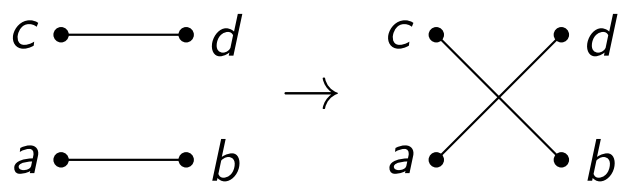
SHORTEST GRAPH TRANSFORMATION (SGT)

INPUT: Two multigraphs G,H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H



$(ab, cd) \to (ac, bd)$



$(ab, cd) \to (ad, bc)$

Shortest Connected Graph Transformation

Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

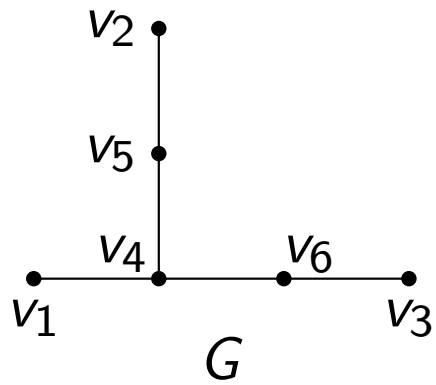
OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity

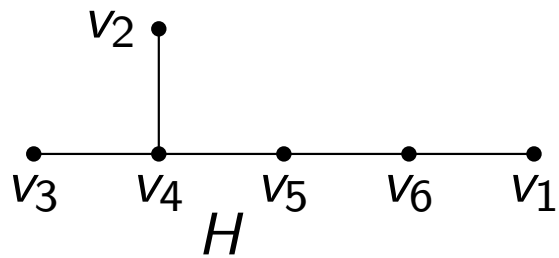
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



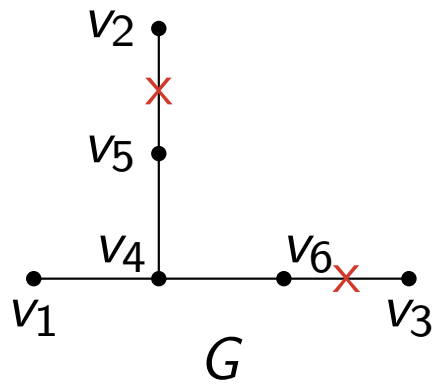


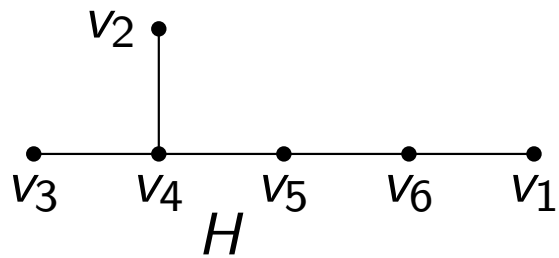
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



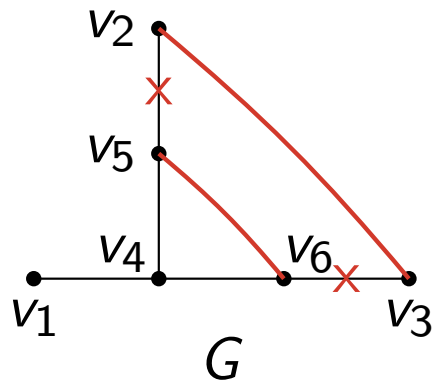


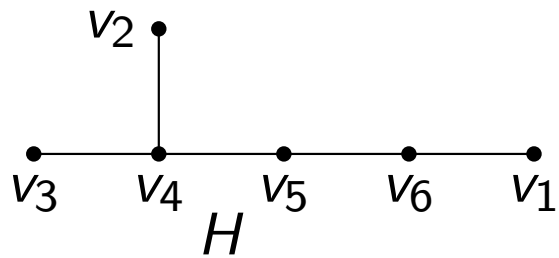
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



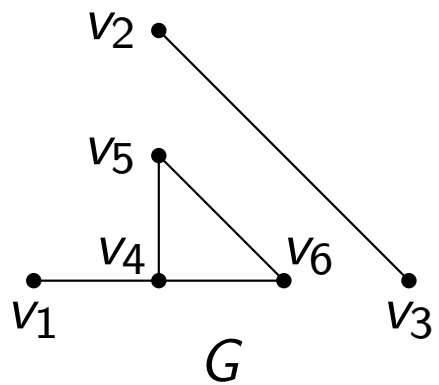


Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

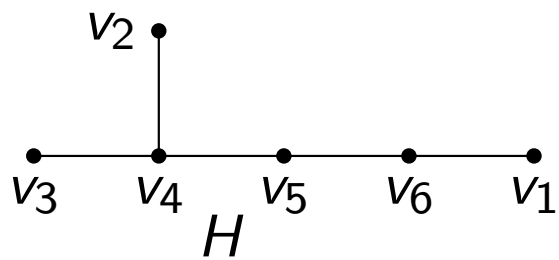
INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



G

DISCONNECTED



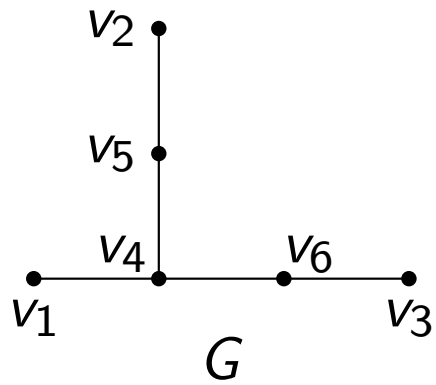
H

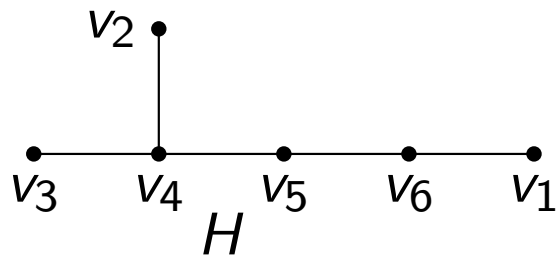
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



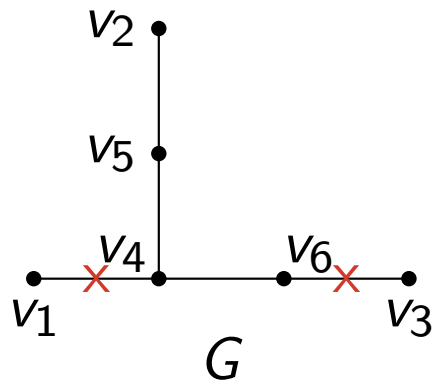


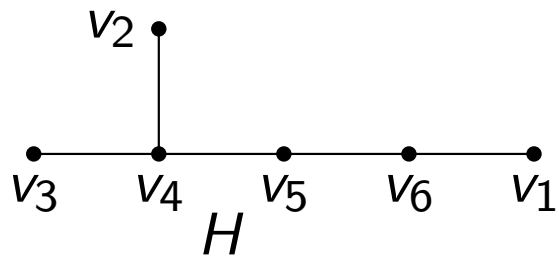
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



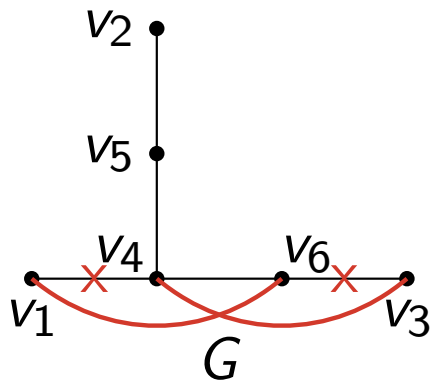


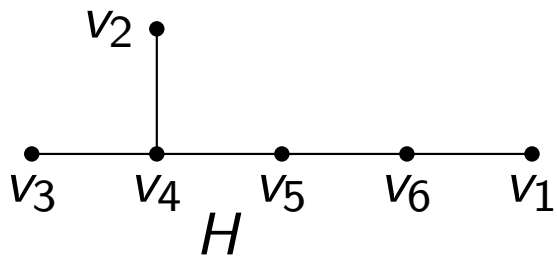
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



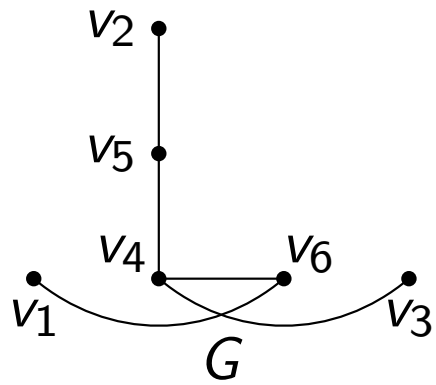


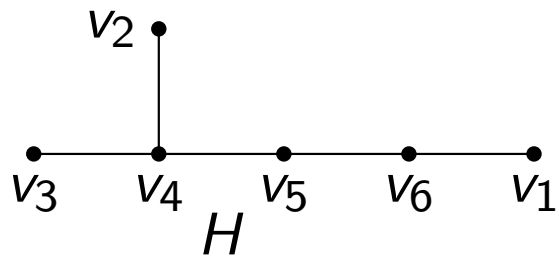
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



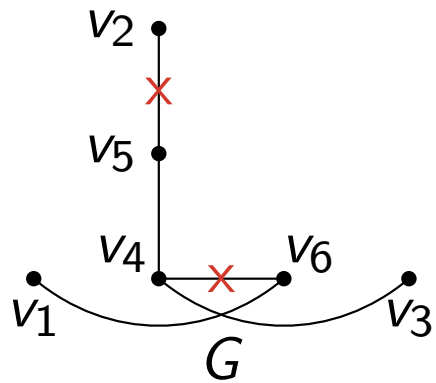


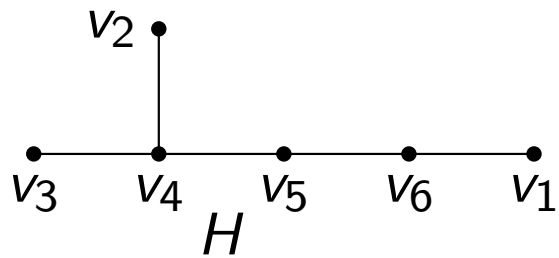
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



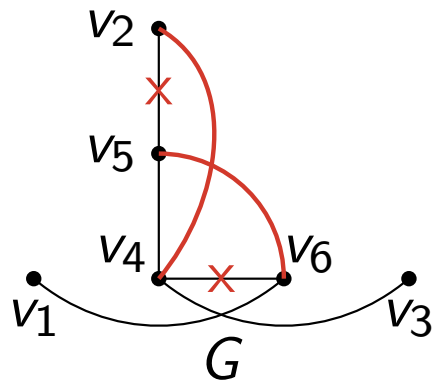


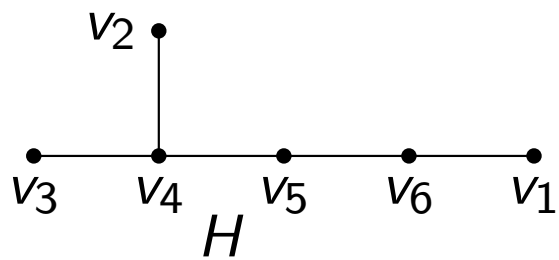
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity



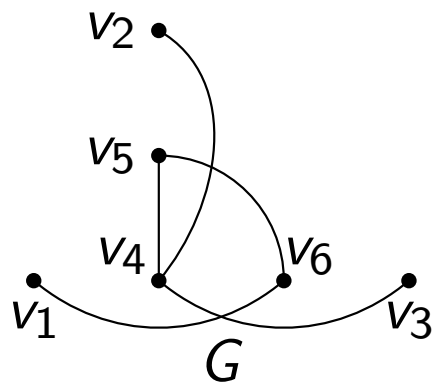


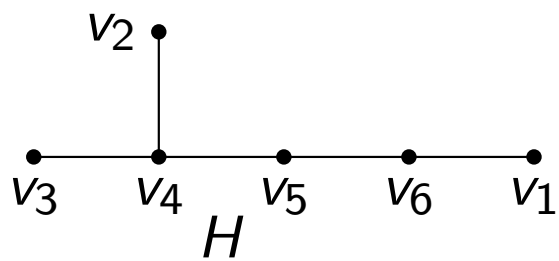
Shortest Connected Graph Transformation

SHORTEST CONNECTED GRAPH TRANSFORMATION (SCGT)

INPUT: Two connected multigraphs G, H with the same vertices and the same degree sequence, an integer k

OUTPUT: True iff there exists a sequence of at most k *flips* that transforms G into H maintaining connectivity





Sorting by Reversals

Sorting by Reversals

SORTING BY REVERSALS (SBR)

Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k

Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P,P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P,P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

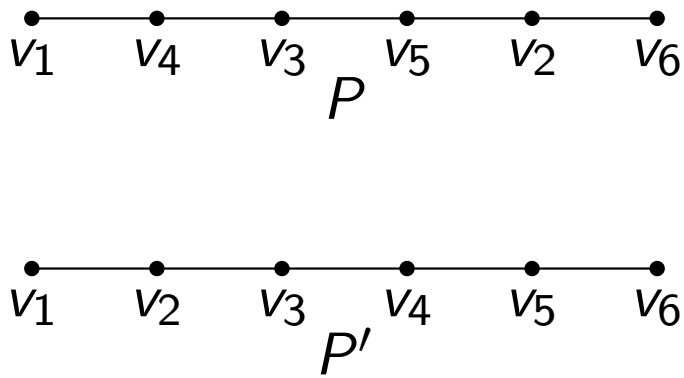
Reversal: Inversion of a subpath

Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

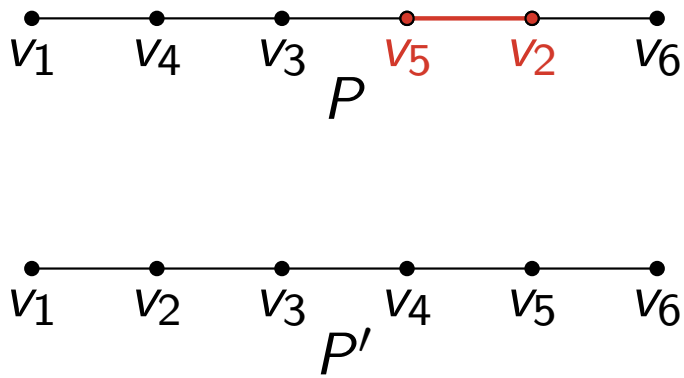


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

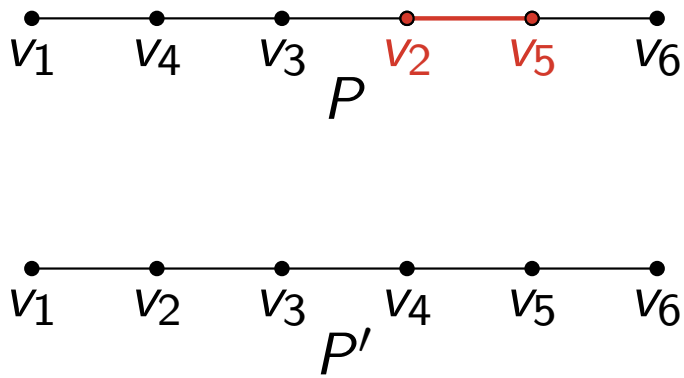


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

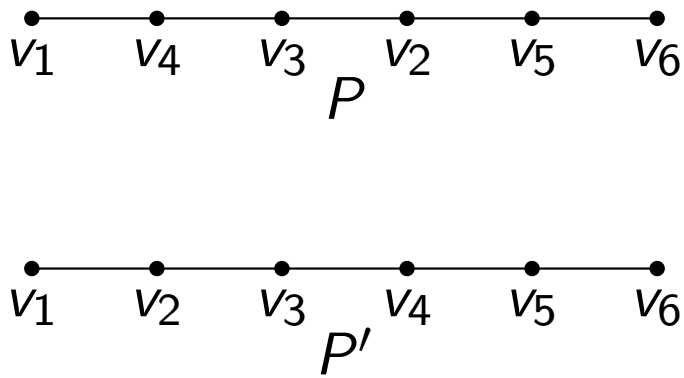


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

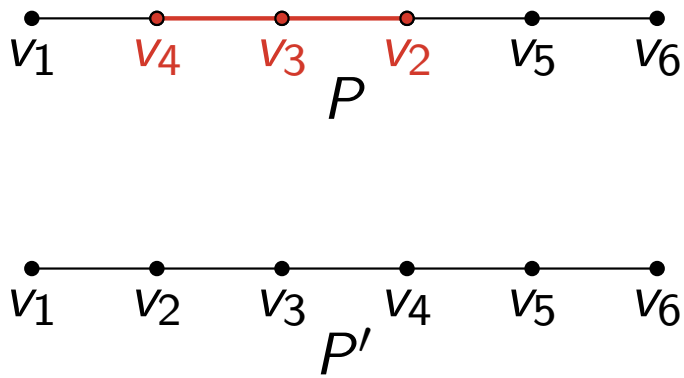


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

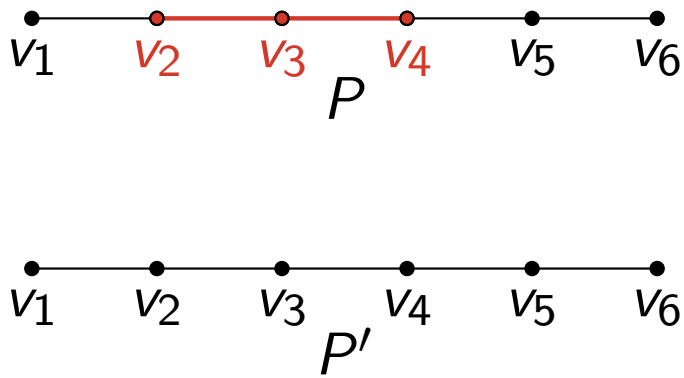


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

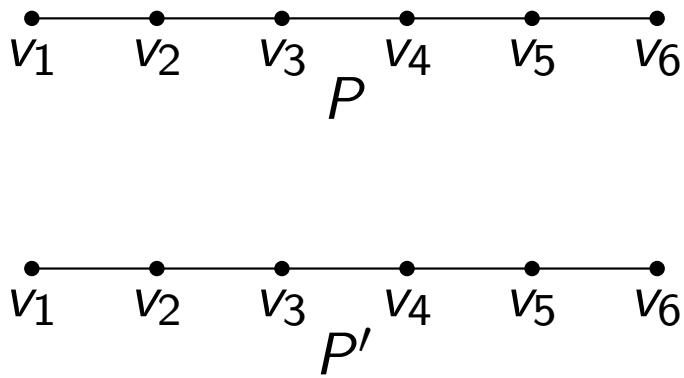


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath

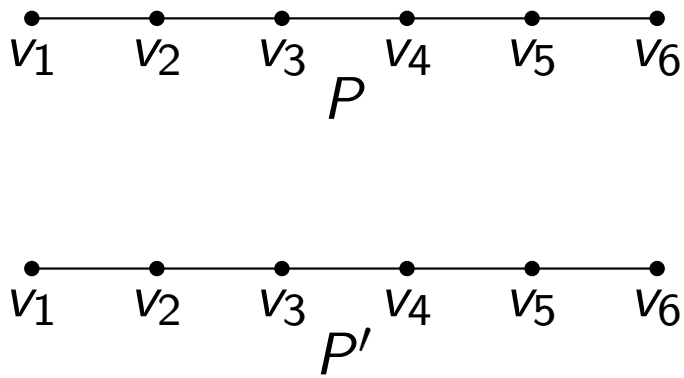


Sorting by Reversals

SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath



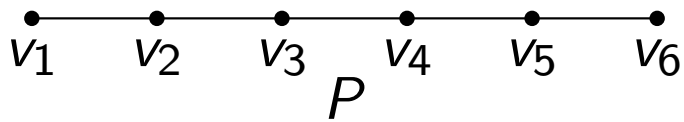
- For paths, SCGT is equivalent to SbR

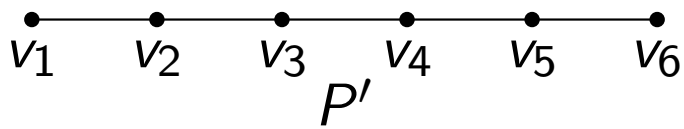
Sorting by Reversals

SORTING BY REVERSALS (SBR)

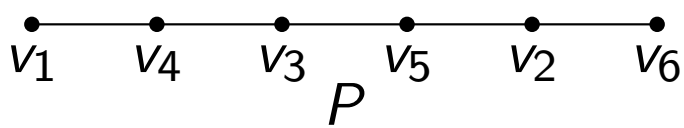
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR

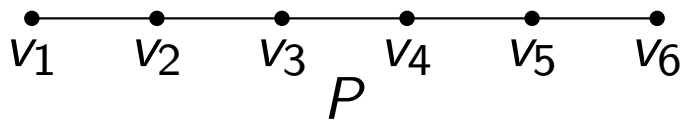


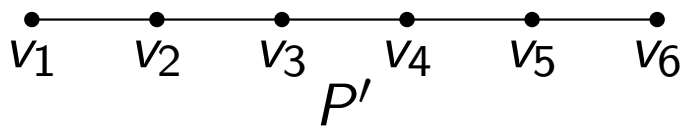
Sorting by Reversals

SORTING BY REVERSALS (SBR)

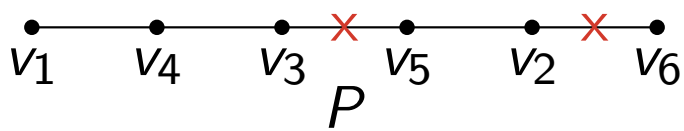
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR

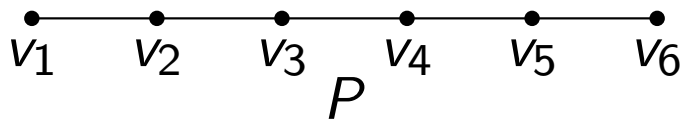


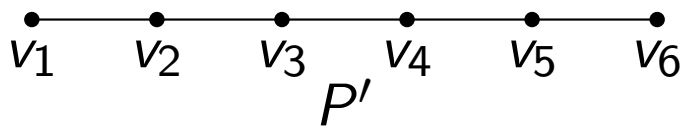
Sorting by Reversals

SORTING BY REVERSALS (SBR)

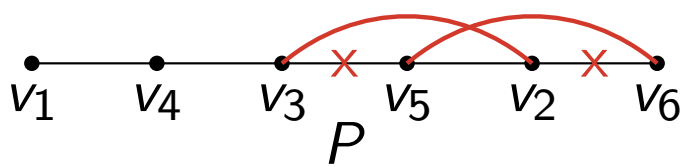
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR

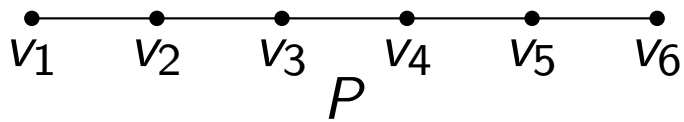


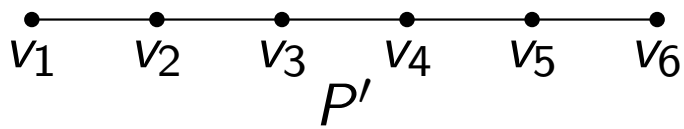
Sorting by Reversals

SORTING BY REVERSALS (SBR)

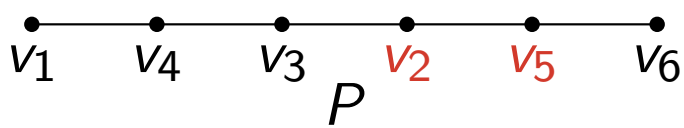
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR

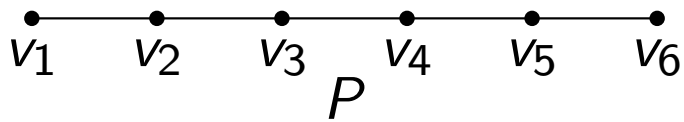


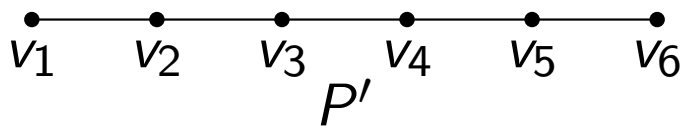
Sorting by Reversals

SORTING BY REVERSALS (SBR)

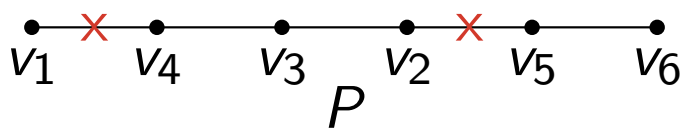
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR

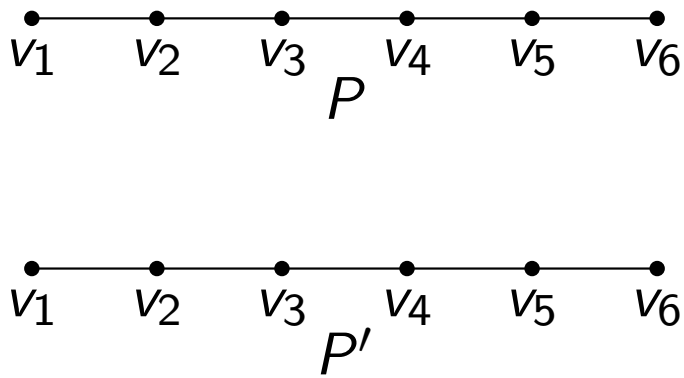


Sorting by Reversals

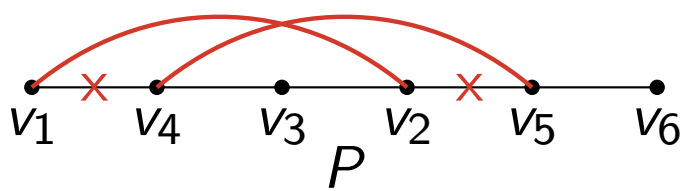
SORTING BY REVERSALS (SBR)

INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath



- For paths, SCGT is equivalent to SbR

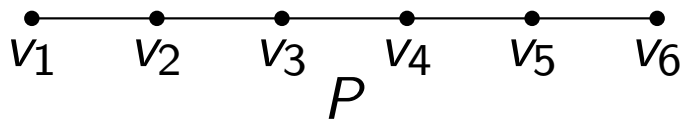


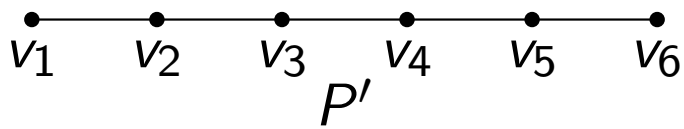
Sorting by Reversals

SORTING BY REVERSALS (SBR)

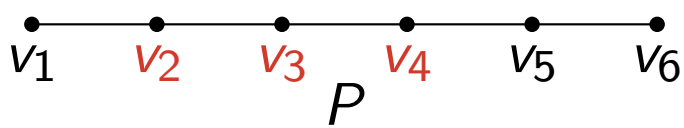
INPUT: Two paths P, P' with same vertices and leaves, an integer k
OUTPUT: True iff there exists a sequence of at most k reversals that transforms P into P'

Reversal: Inversion of a subpath





- For paths, SCGT is equivalent to SbR



Applications

Applications

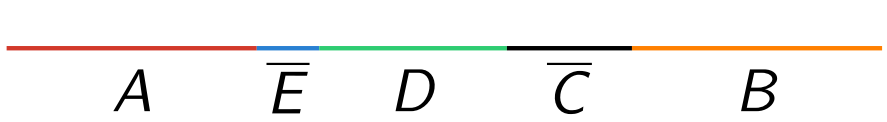
- Application of SbR to the computation of genetic distance between species

Applications

- Application of SbR to the computation of genetic distance between species

Applications

- Application of SbR to the computation of genetic distance between species

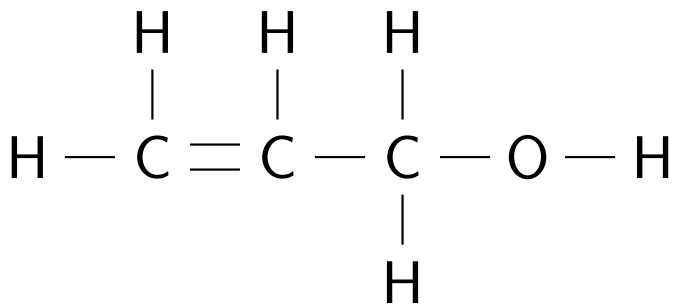


- Application of SCGT to mass spectrometry

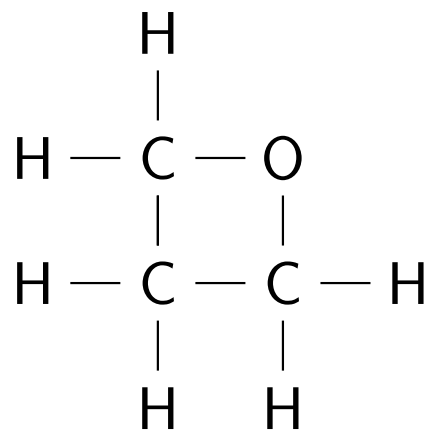
Applications

- Application of SbR to the computation of genetic distance between species

- Application of SCGT to mass spectrometry



allyl alcohol C_3H_6O



oxetane C_3H_6O

State of the Art

State of the Art

- SbR:

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:
  - Hakimi, 63: Always a transformation for SGT

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:
  - Hakimi, 63: Always a transformation for SGT
  - Will, 99: SGT is NP-complete

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:
  - Hakimi, 63: Always a transformation for SGT
  - Will, 99: SGT is NP-complete
  - Bereg and Ito, 16: 1.5-approximation algorithm for SGT

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:
  - Hakimi, 63: Always a transformation for SGT
  - Will, 99: SGT is NP-complete
  - Bereg and Ito, 16: 1.5-approximation algorithm for SGT
  - Taylor, 81: Always a transformation for SCGT

State of the Art

- SbR:
  - Capraca, 97: SbR is NP-complete
  - Berman et al, 02: 1.375-approximation algorithm for SbR
- SGT and SCGT:
  - Hakimi, 63: Always a transformation for SGT
  - Will, 99: SGT is NP-complete
  - Bereg and Ito, 16: 1.5-approximation algorithm for SGT
  - Taylor, 81: Always a transformation for SCGT
  - Bousquet and Mary, 18: 4-approximation algorithm for SCGT

Preliminaries

Preliminaries

- *Distance* $d(G, H)$ ($d_c(G, H)$): length of a shortest (connected) transformation between G and H

Preliminaries

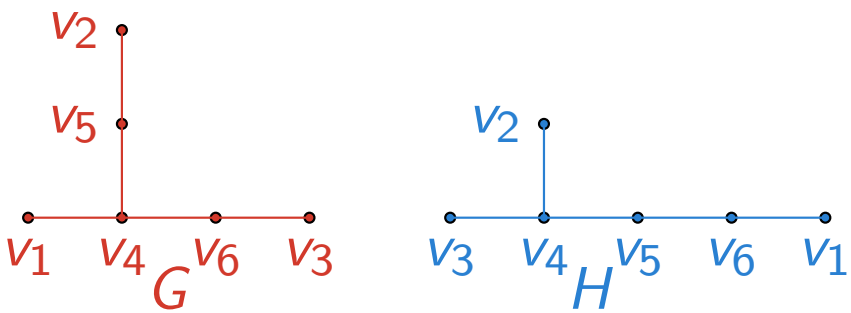
- *Distance* $d(G, H)$ ($d_c(G, H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G, H)$

Preliminaries

- *Distance* $d(G, H)$ ($d_c(G, H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G, H)$
- *Symmetric difference* $\Delta(G, H)$: $(G - H) \cup (H - G)$

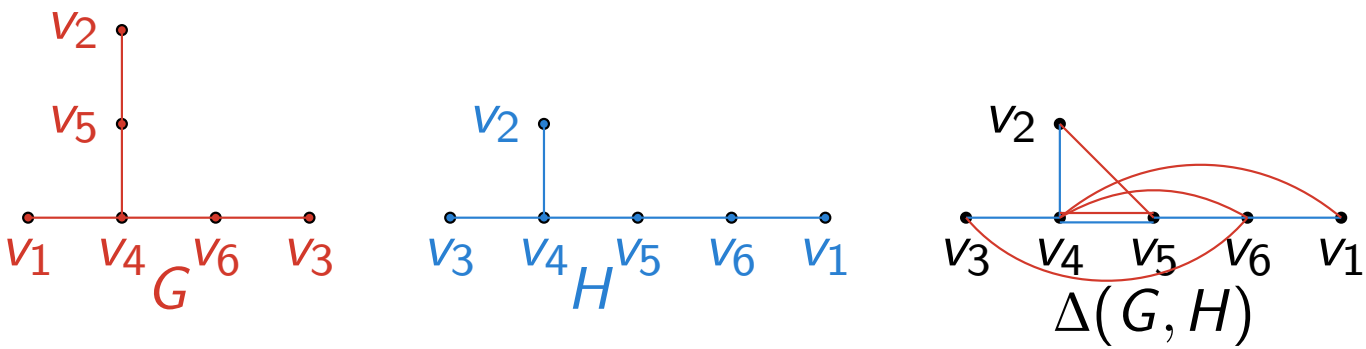
Preliminaries

- *Distance* $d(G, H)$ ($d_c(G, H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G, H)$
- *Symmetric difference* $\Delta(G, H)$: $(G - H) \cup (H - G)$



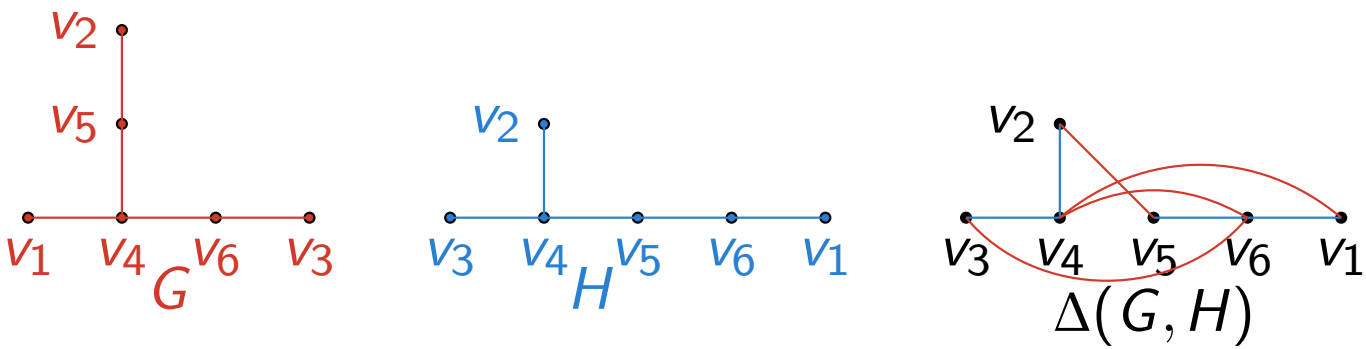
Preliminaries

- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



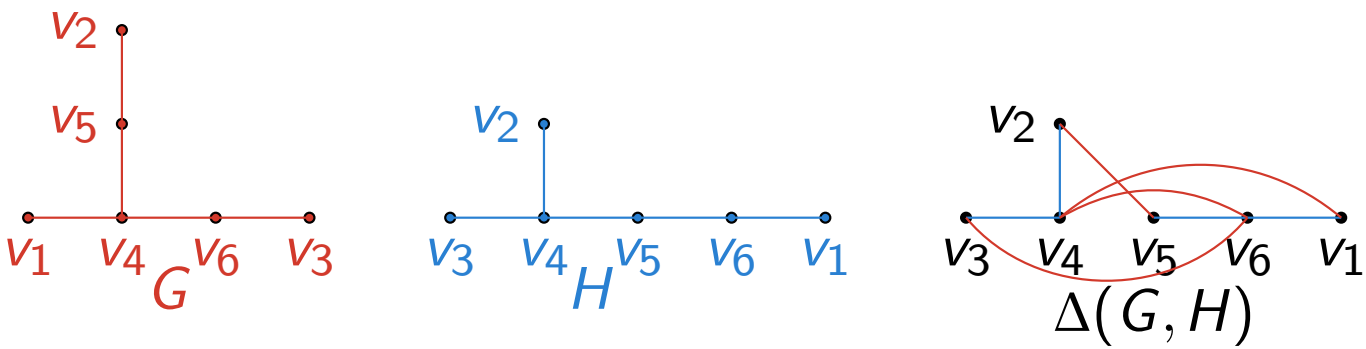
Preliminaries

- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



Preliminaries

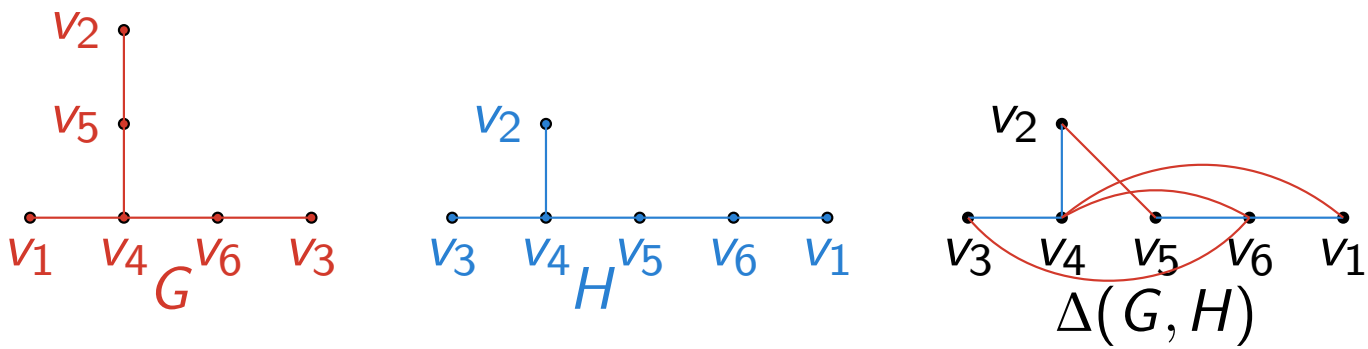
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$

Preliminaries

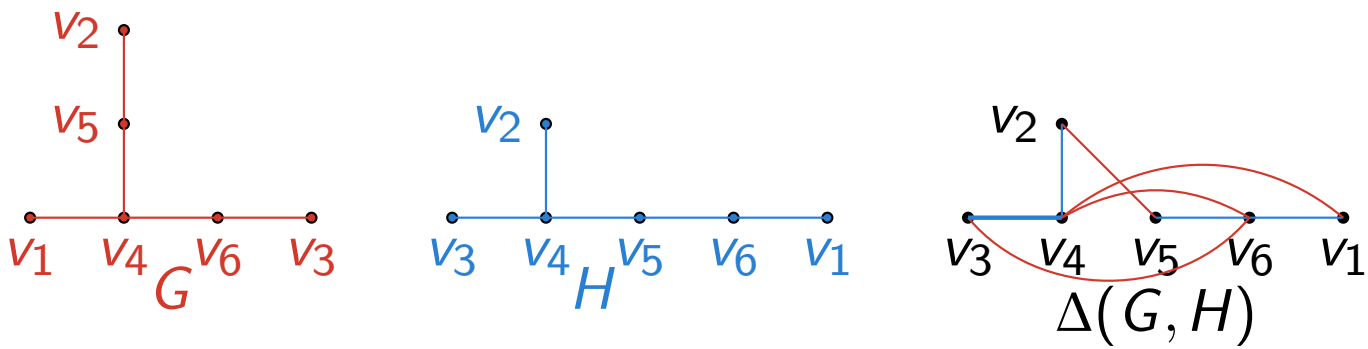
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

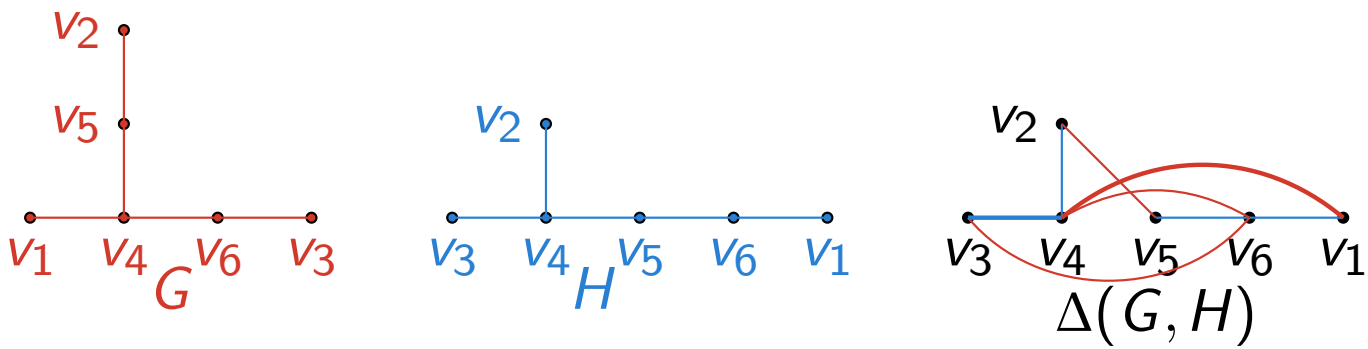
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

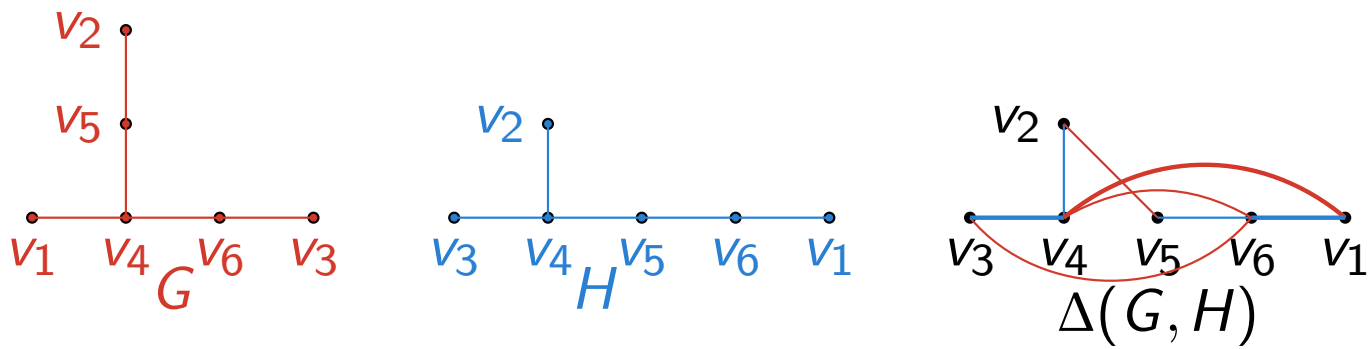
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

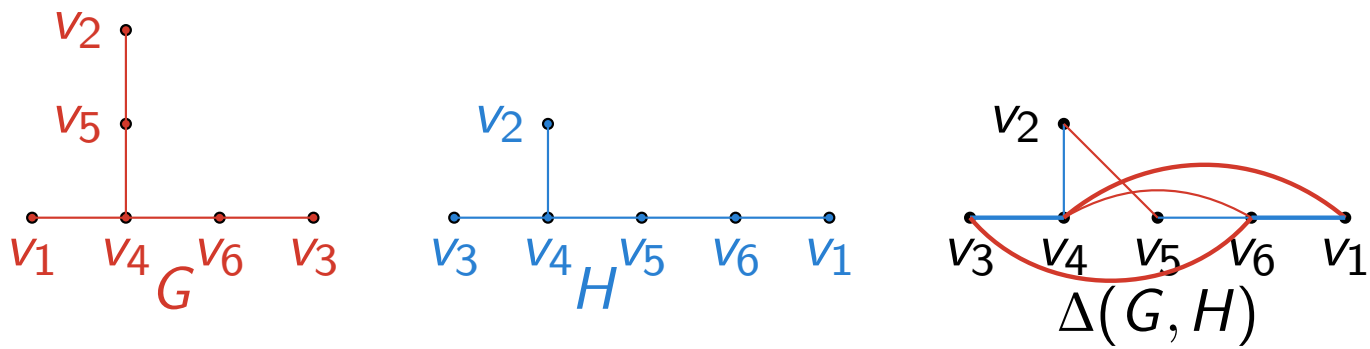
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

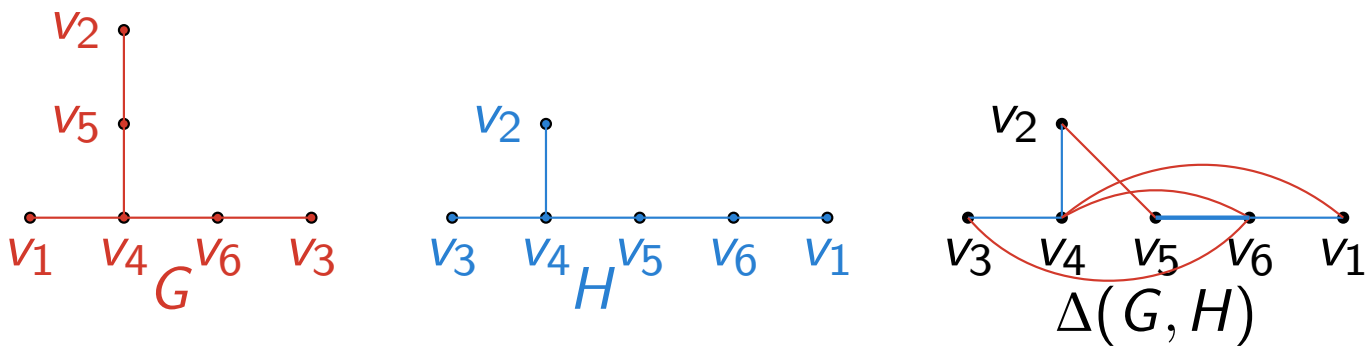
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

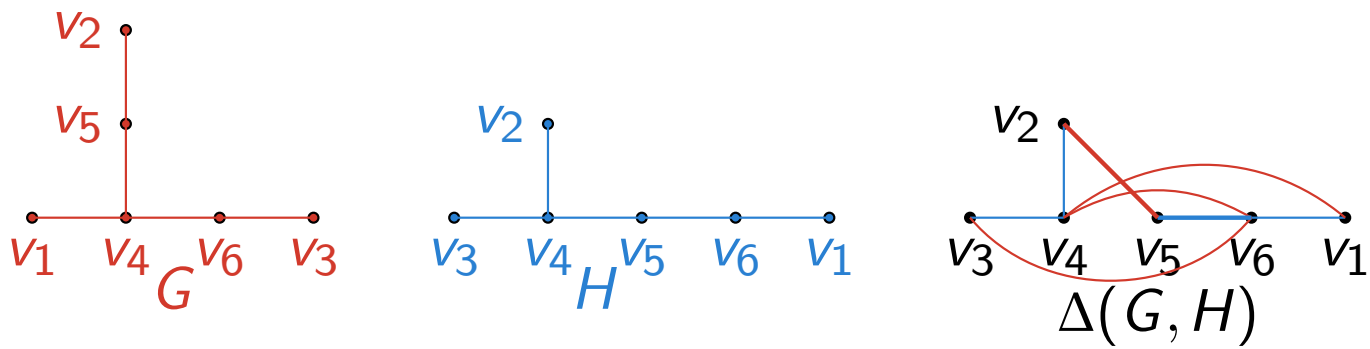
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

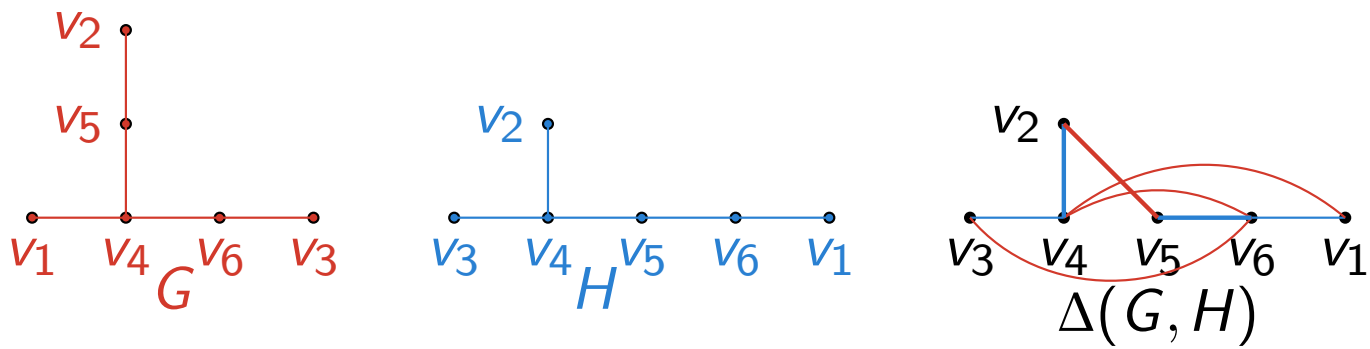
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

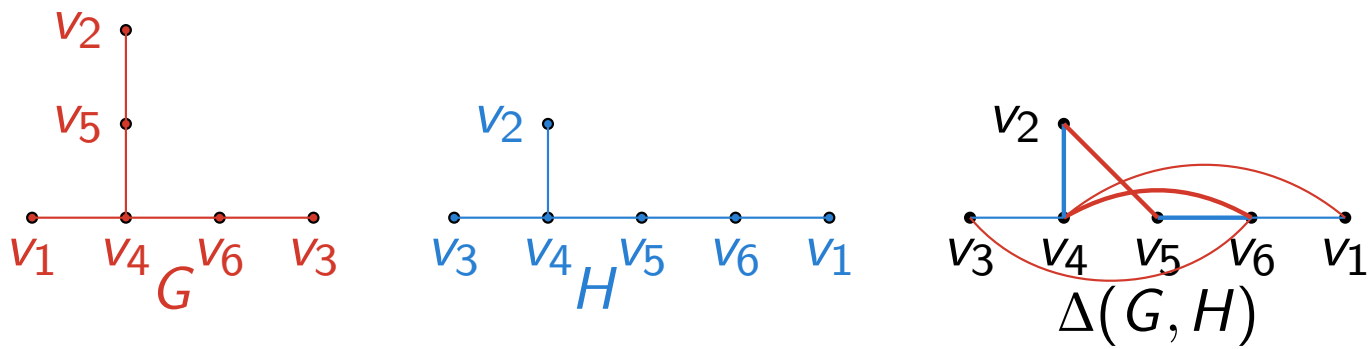
- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

Preliminaries

- *Distance* $d(G,H)$ ($d_c(G,H)$): length of a shortest (connected) transformation between G and H
- Our goal: Approximate $d_c(G,H)$
- *Symmetric difference* $\Delta(G,H)$: $(G-H)\cup(H-G)$



- $d_c(G,H) \geq d(G,H) \geq \frac{|\Delta(G,H)|}{4}$
- Will, 99: $d(G,H) = \frac{|\Delta(G,H)|}{2} - mnc(G,H)$ where $mnc(G,H)$ is the maximum number of cycles in a partition of $\Delta(G,H)$ into alternating cycles.

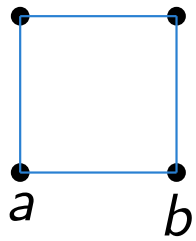
Why trees ?

Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce$|\Delta(G, H)|$ by 2 in 1 flip

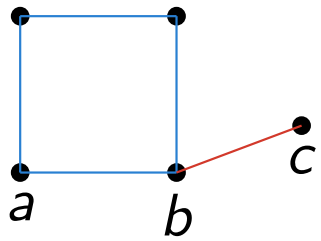
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



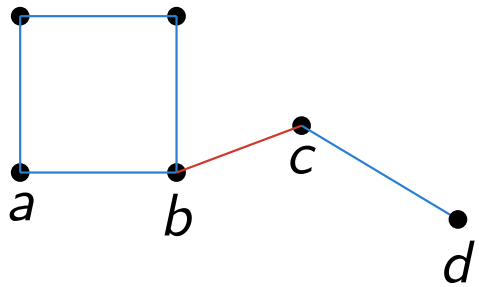
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



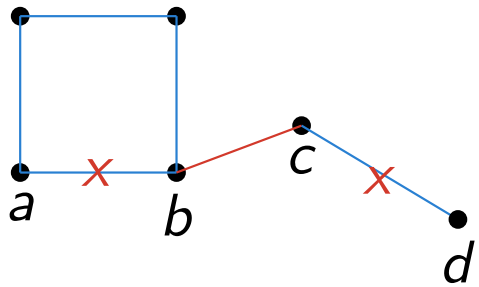
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



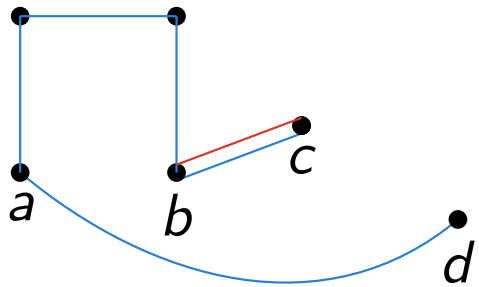
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



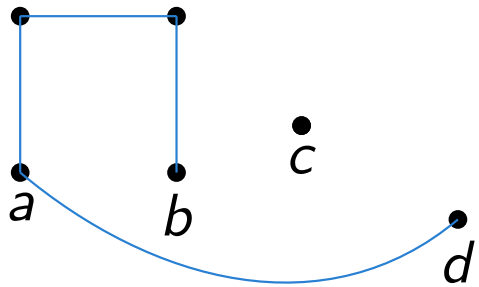
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



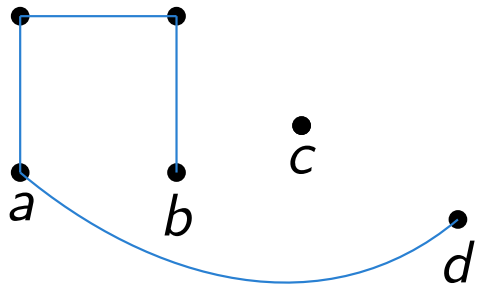
Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



Why trees ?

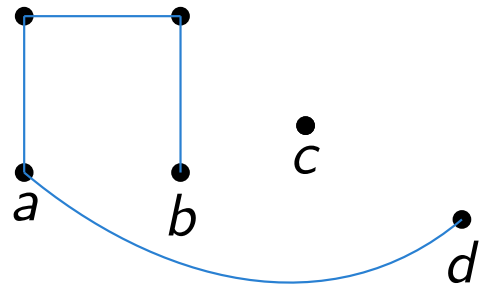
- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



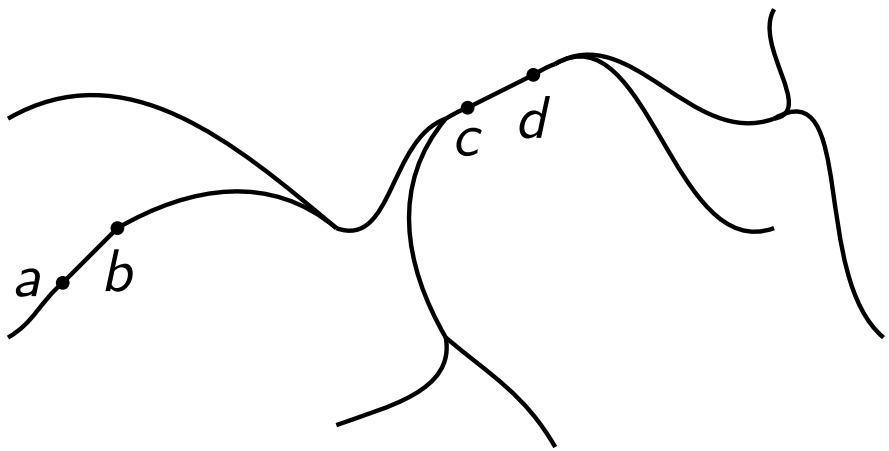
- Consequences of a flip on a tree:

Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip

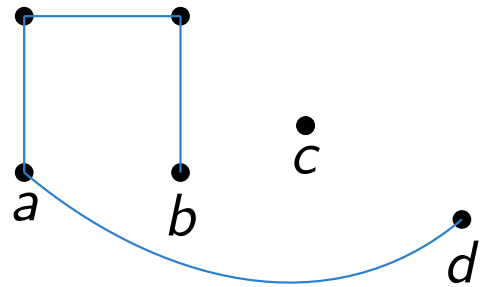


- Consequences of a flip on a tree:

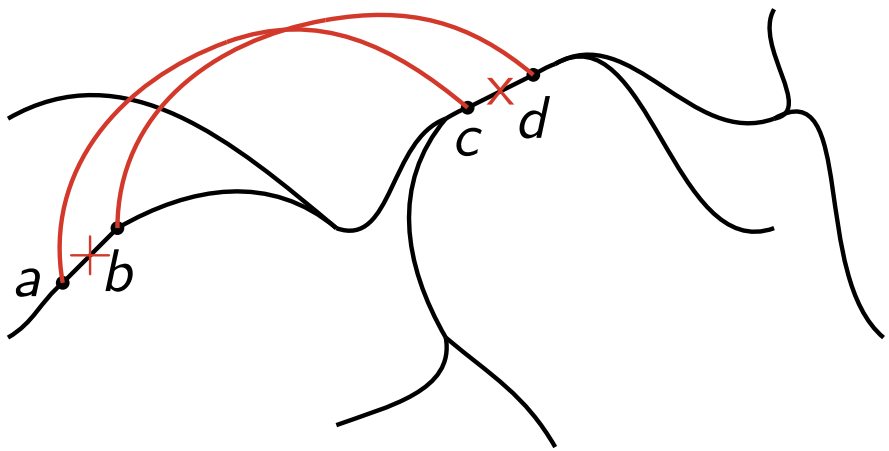


Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip

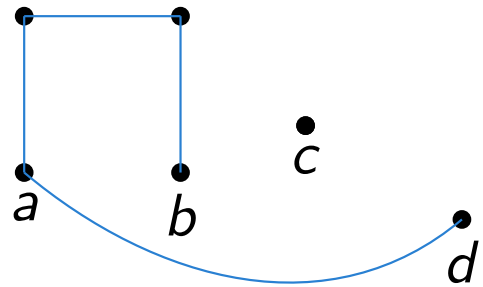


- Consequences of a flip on a tree:

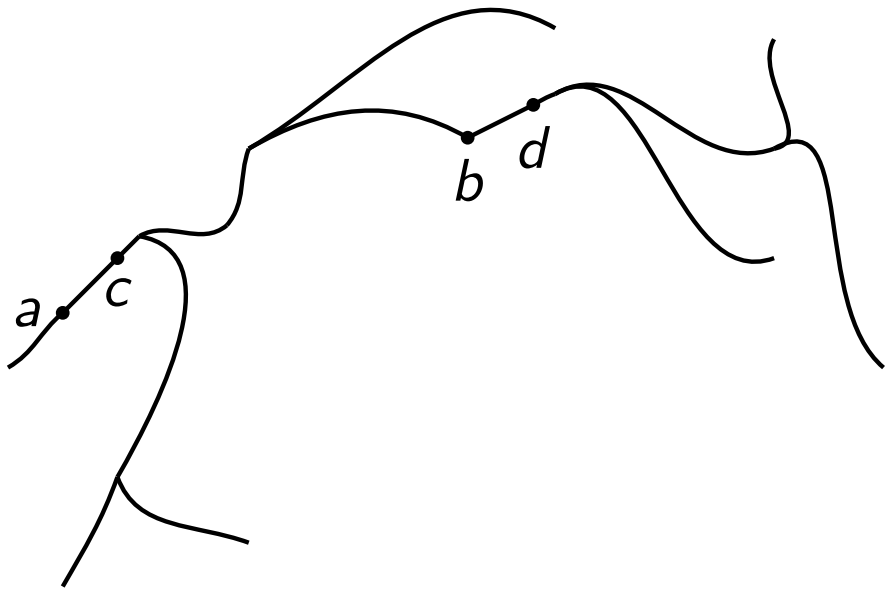


Why trees ?

- If an edge ab of $G - H$ is in a cycle of G, we reduce $|\Delta(G, H)|$ by 2 in 1 flip



- Consequences of a flip on a tree:



Our main result

Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

- Upper bound: Reduce$|\Delta(G, H)|$ by at least 4 in at most 3 flips

Our main result

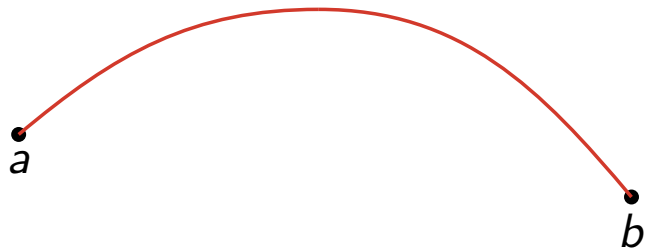
Theorem: SCGT admits a 2.5-approximation algorithm.

- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips

Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

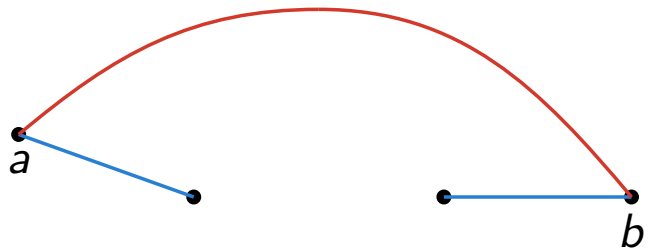
- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

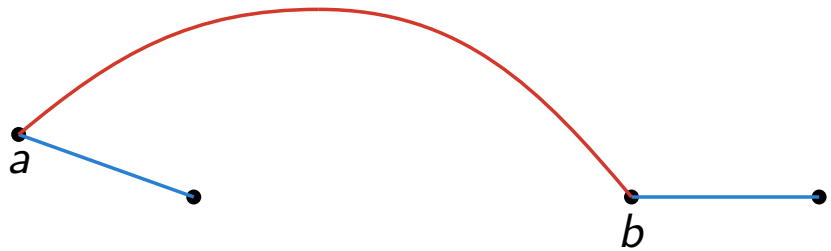
- Upper bound: Reduce$|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

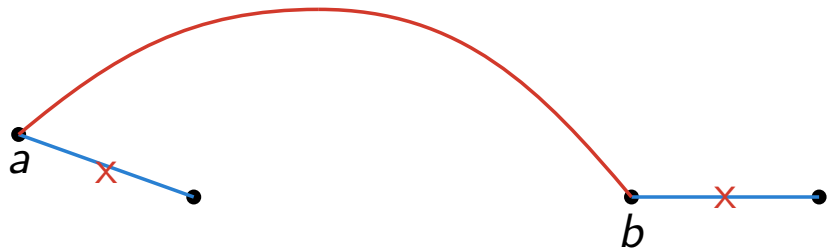
- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

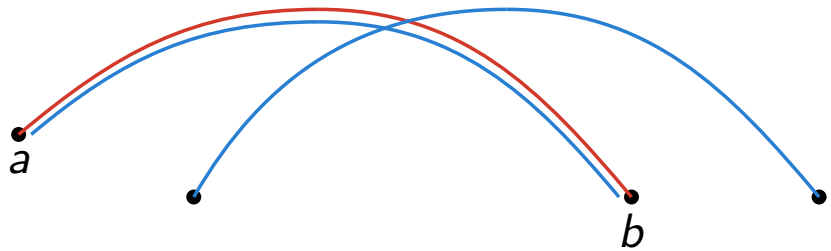
- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

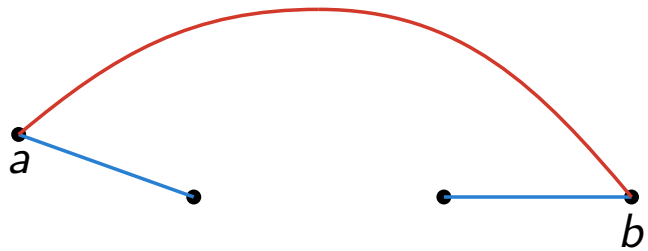
- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

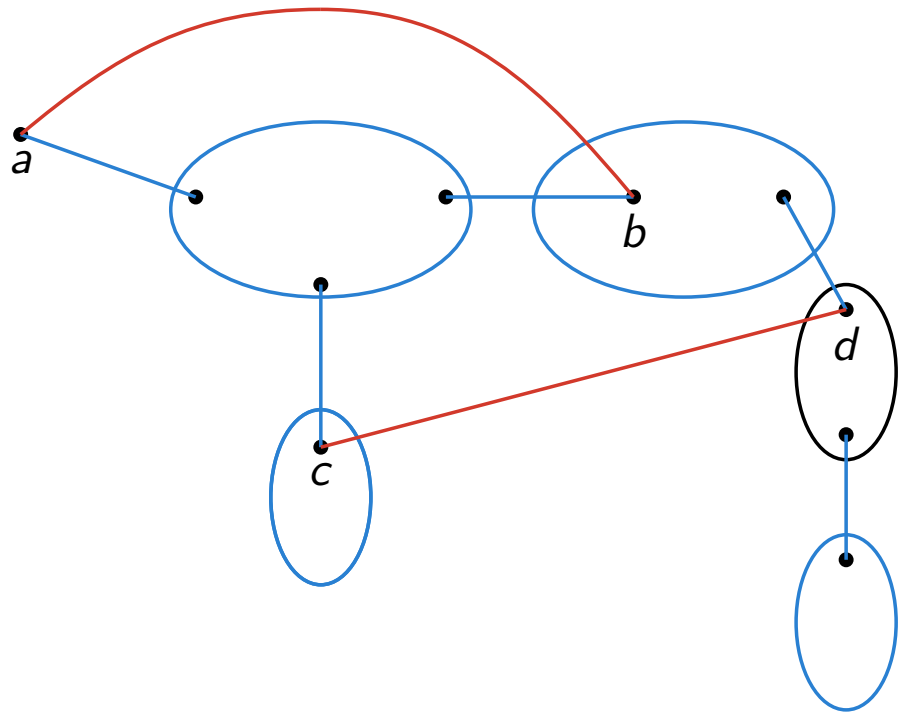
- Upper bound: Reduce$|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

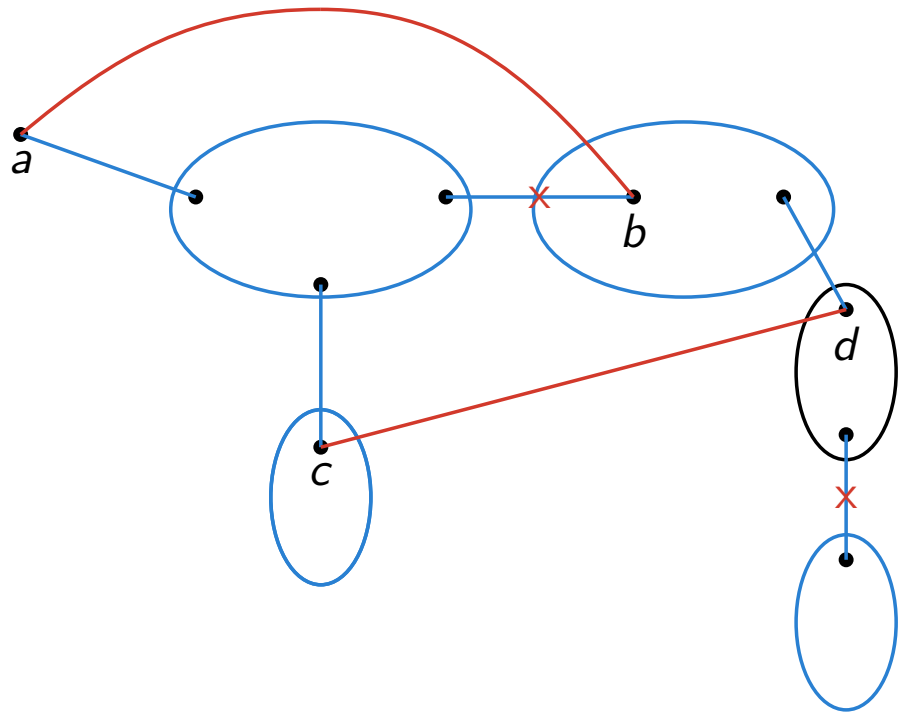
- Upper bound: Reduce$|\Delta(G,H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

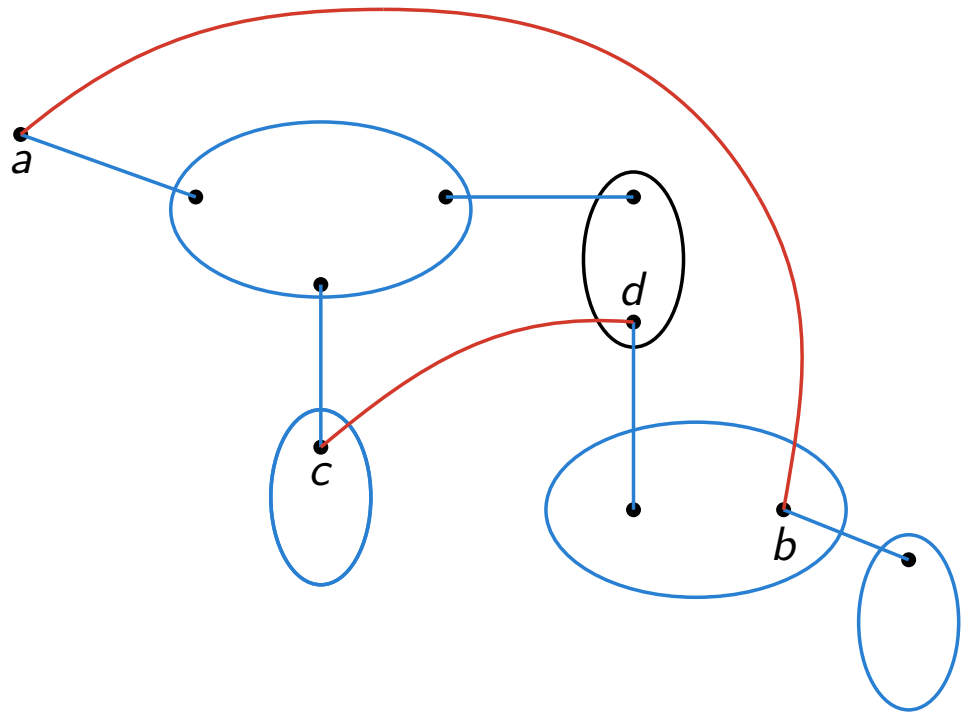
- Upper bound: Reduce$|\Delta(G,H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

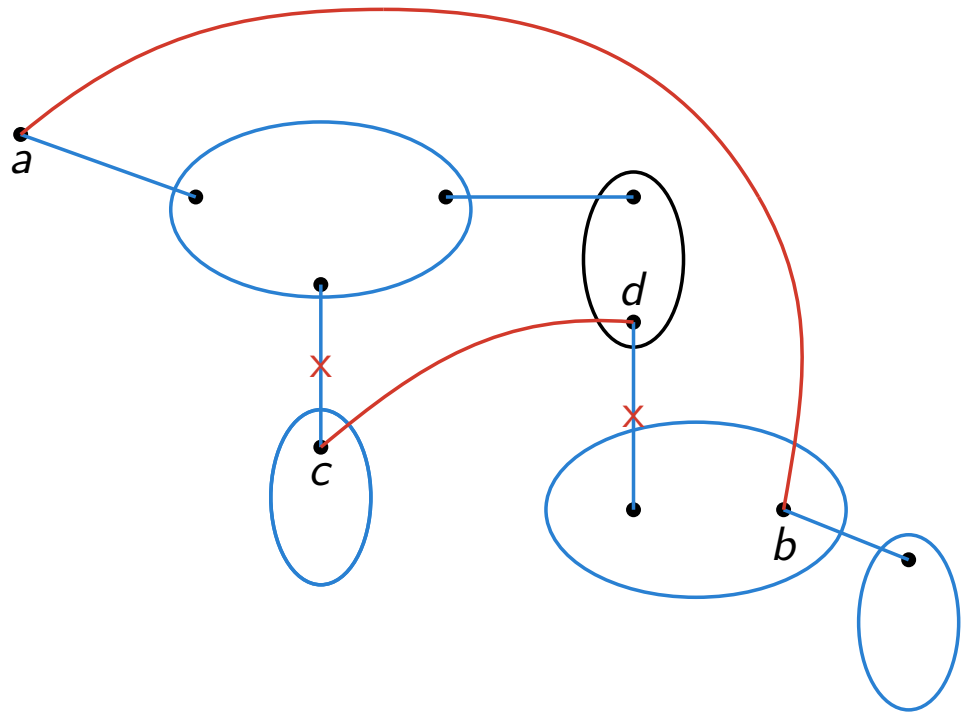
- Upper bound: Reduce $|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

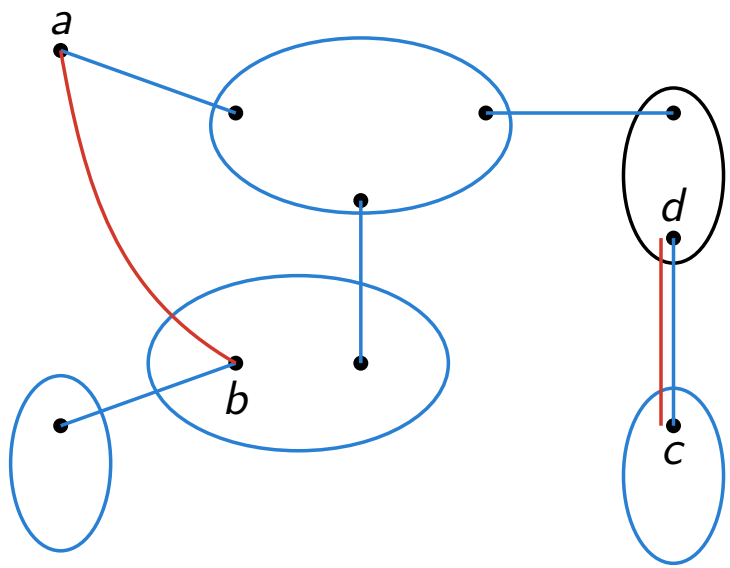
- Upper bound: Reduce$|\Delta(G,H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

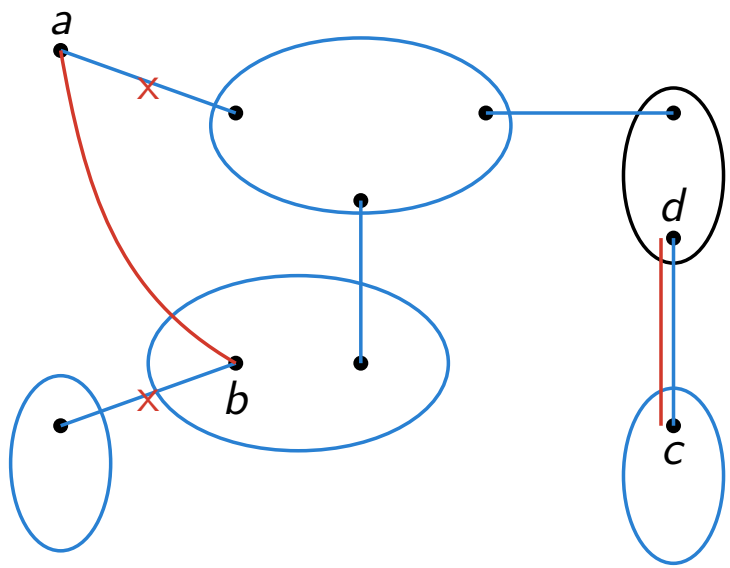
- Upper bound: Reduce$|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

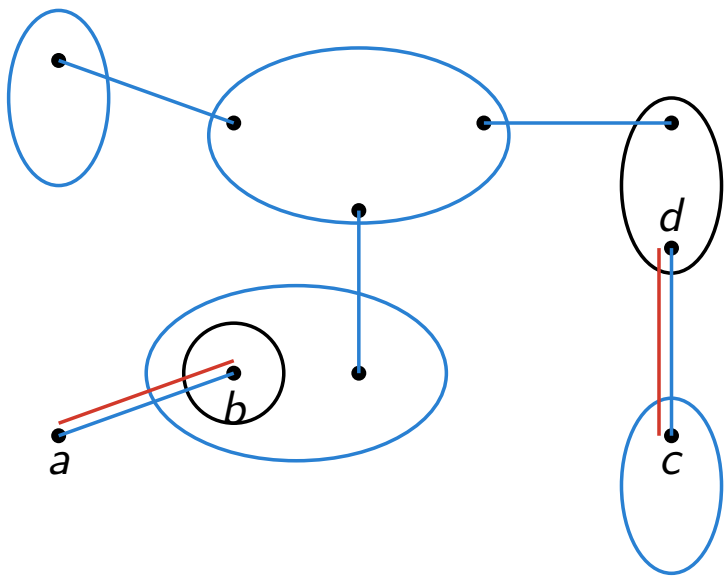
- Upper bound: Reduce$|\Delta(G,H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

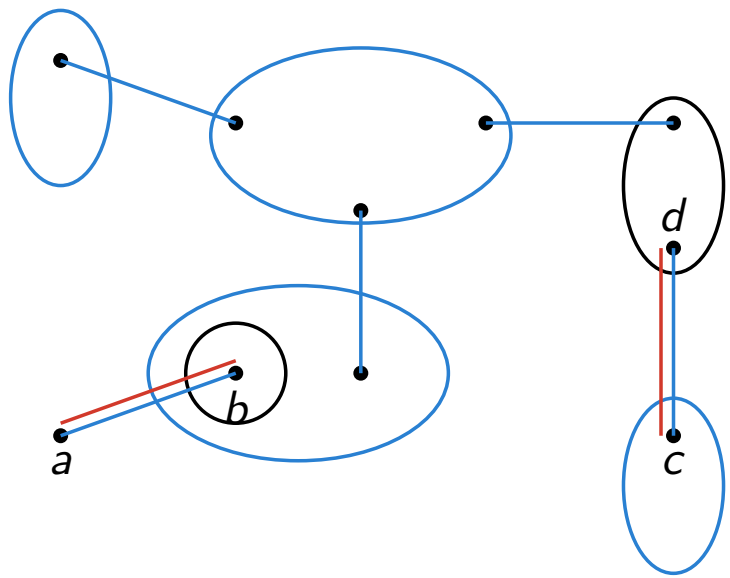
- Upper bound: Reduce$|\Delta(G, H)|$ by at least 4 in at most 3 flips



Our main result

Theorem: SCGT admits a 2.5-approximation algorithm.

- Upper bound: Reduce$|\Delta(G,H)|$ by at least 4 in at most 3 flips



- Lower bound: $d_c(G,H) \geq \frac{|\Delta(G,H)|}{2} - mnc(G,H)$

Discussion about the lower bound

Discussion about the lower bound

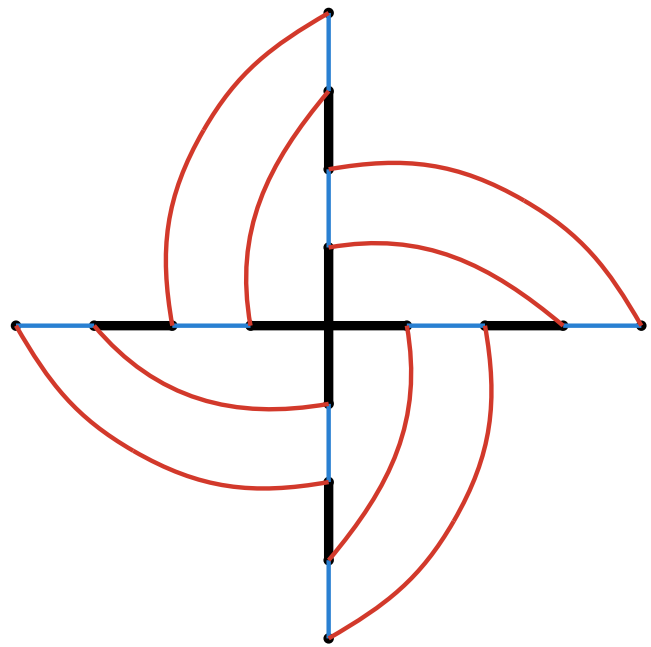
- There exist G_k and H_k for which if we only flip edges of the symmetric difference,

$$d_c(G_k, H_k) \geq \frac{3}{2}(\frac{|\Delta(G_k, H_k)|}{2} - mnc(G_k, H_k))$$

Discussion about the lower bound

- There exist G_k and H_k for which if we only flip edges of the symmetric difference,

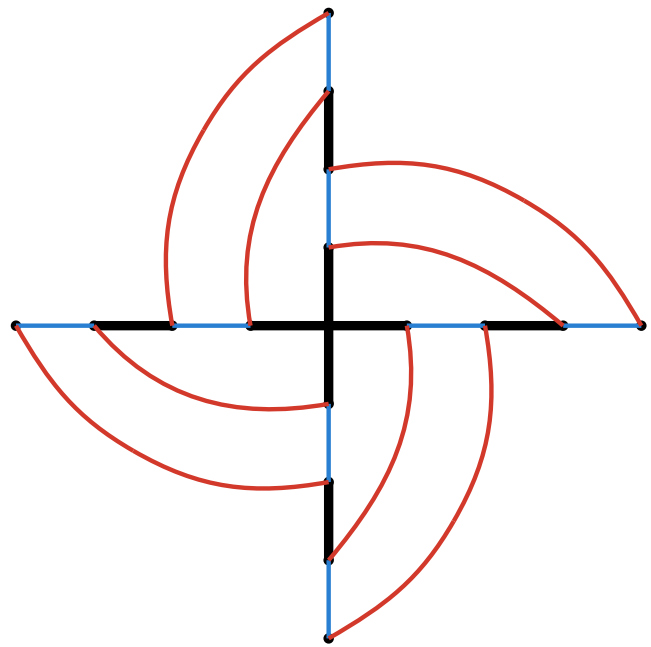
$$d_c(G_k, H_k) \geq \frac{3}{2}\left(\frac{|\Delta(G_k, H_k)|}{2} - mnc(G_k, H_k)\right)$$



Discussion about the lower bound

- There exist G_k and H_k for which if we only flip edges of the symmetric difference,

$$d_c(G_k, H_k) \geq \frac{3}{2}\left(\frac{|\Delta(G_k, H_k)|}{2} - mnc(G_k, H_k)\right)$$



- With this lower bound, can not do better than a 1.5-approximation

Open questions

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$
  $\rightarrow$ True when we only flip edges of the same cycle of the decomposition of $\Delta(G, H)$ into alternating cycles

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$
  $\rightarrow$ True when we only flip edges of the same cycle of the decomposition of $\Delta(G,H)$ into alternating cycles
- A shortest transformation from G to H only flips edges of $\Delta(G,H)$?

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$
  $\rightarrow$ True when we only flip edges of the same cycle of the decomposition of $\Delta(G,H)$ into alternating cycles
- A shortest transformation from G to H only flips edges of $\Delta(G,H)$?
  $\rightarrow$ Open for paths

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$
  $\rightarrow$ True when we only flip edges of the same cycle of the decomposition of $\Delta(G, H)$ into alternating cycles
- A shortest transformation from G to H only flips edges of $\Delta(G, H)$?
  $\rightarrow$ Open for paths
- Find a better lower bound to improve our approximation ratio

Open questions

- Conjecture: The shortest transformation from G_k to H_k has length $\frac{|\Delta(G_k,H_k)|}{2} - 1$
  $\rightarrow$ True when we only flip edges of the same cycle of the decomposition of $\Delta(G,H)$ into alternating cycles
- A shortest transformation from G to H only flips edges of $\Delta(G,H)$?
  $\rightarrow$ Open for paths
- Find a better lower bound to improve our approximation ratio