Order Reconfiguration under Width Constraints

Emmanuel Arrighi[1], Henning Fernau[2], Mateus de Oliveira Oliveira[1], Petra Wolf[2].

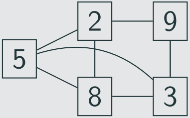
BIRS Reconfiguration Workshop 22w5090 May 10th, 2022
also see: MFCS 2021

[1] University of Bergen, Norway
[2] University of Trier, Germany

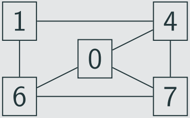
Overview

Graph isomorphism

G_1 :

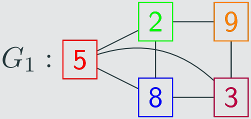


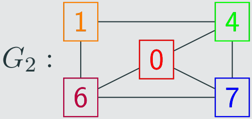
G_2 :



Overview

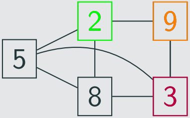
Graph isomorphism

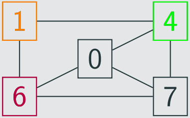




Overview

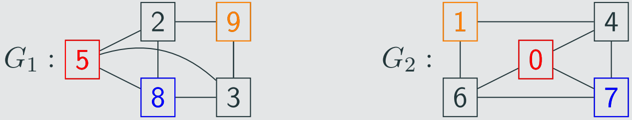
Graph isomorphism

G_1 :


G_2 :


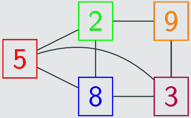
Overview

Graph isomorphism

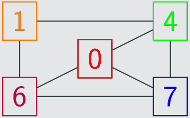


Graph isomorphism

G_1 :

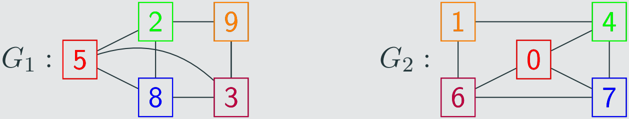


G_2 :



Overview

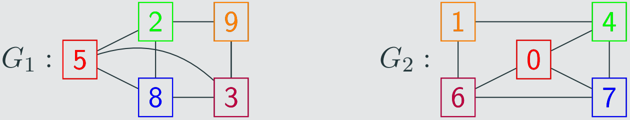
Graph isomorphism



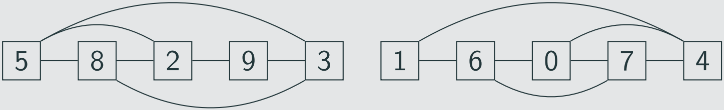
Reconfiguration

Overview

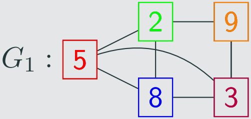
Graph isomorphism

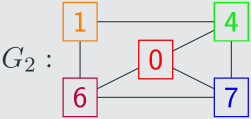


Reconfiguration

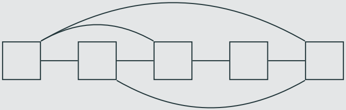


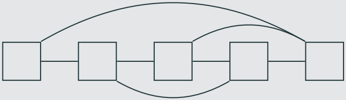
Graph isomorphism





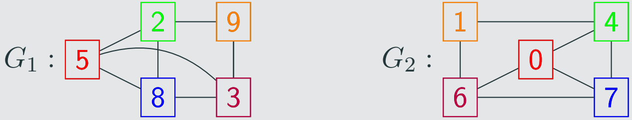
Reconfiguration



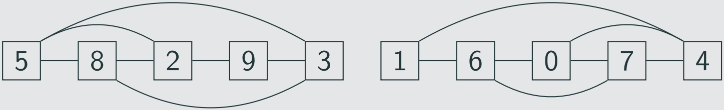


Overview

Graph isomorphism

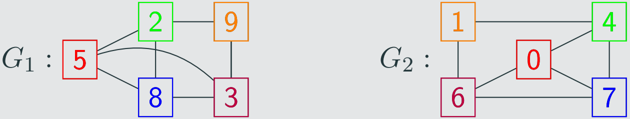


Reconfiguration

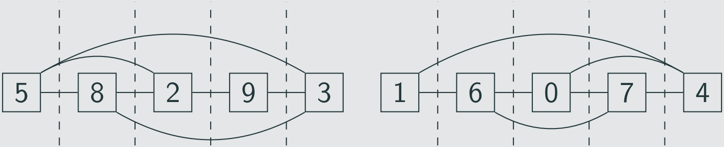


Overview

Graph isomorphism

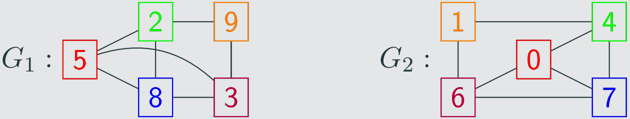


Reconfiguration

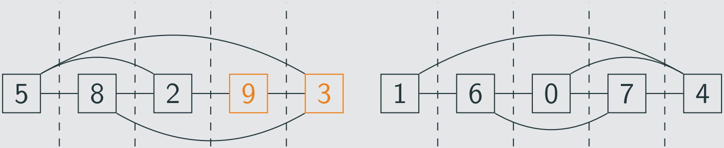


Overview

Graph isomorphism

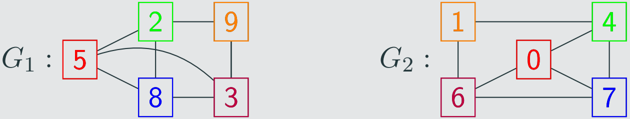


Reconfiguration

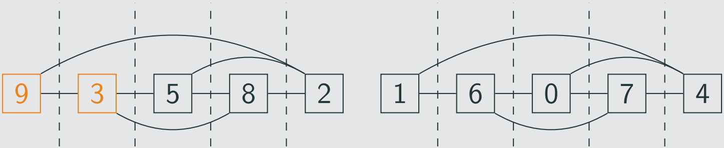


Overview

Graph isomorphism

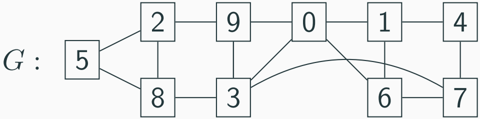


Reconfiguration



Order Reconfiguration

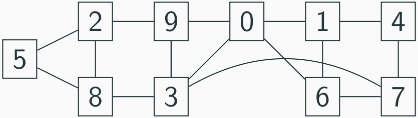
Cutwidth



Definition (Cutwidth of an ordering)

Cutwidth

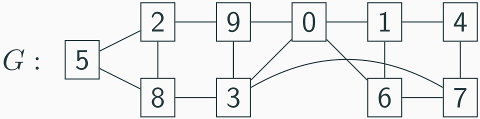
G :



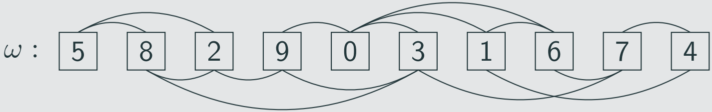
Definition (Cutwidth of an ordering)

ω : 5 8 2 9 0 3 1 6 7 4

Cutwidth

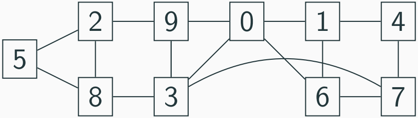


Definition (Cutwidth of an ordering)



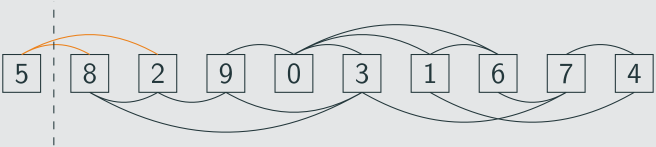
Cutwidth

G :



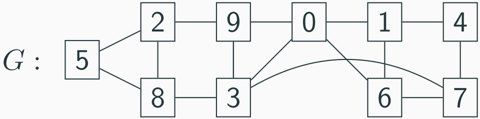
Definition (Cutwidth of an ordering)

ω :

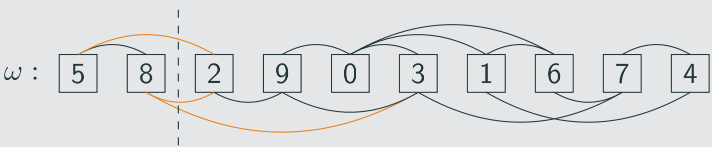


$\mathrm{cw}(G, \omega) = \max\left(\{2,\right.$

Cutwidth

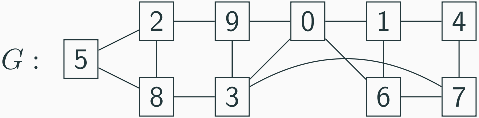


Definition (Cutwidth of an ordering)

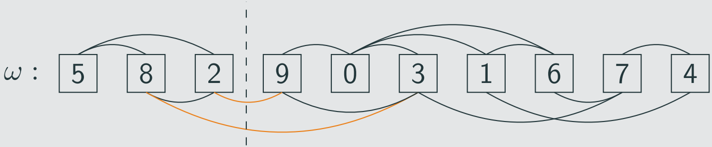


$\mathrm{cw}(G, \omega) = \max\left(\{2, 3,\right.$

Cutwidth

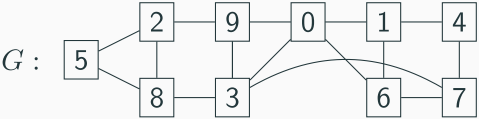


Definition (Cutwidth of an ordering)

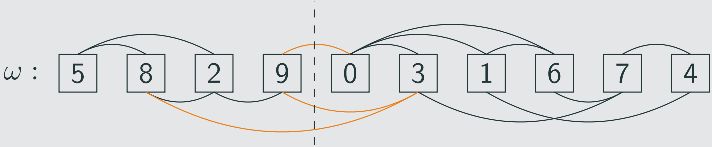


$\mathrm{cw}(G, \omega) = \max\left(\{2, 3, 2,\right.$

Cutwidth

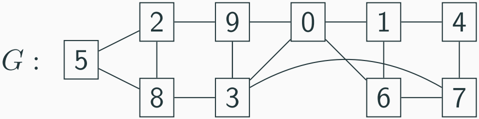


Definition (Cutwidth of an ordering)

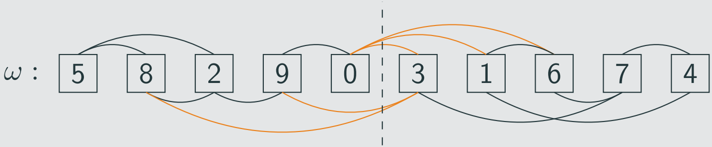


$\mathrm{cw}(G, \omega) = \max\left(\{2, 3, 2, 3,\right.$

Cutwidth

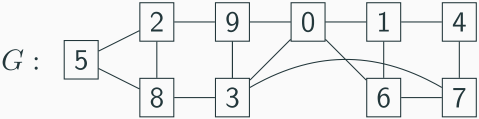


Definition (Cutwidth of an ordering)

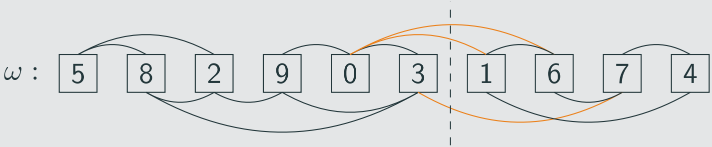


$\mathrm{cw}(G, \omega) = \max(\{2, 3, 2, 3, 5,$

Cutwidth

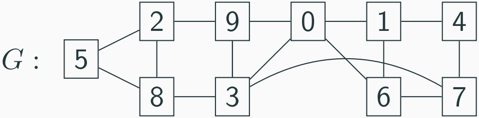


Definition (Cutwidth of an ordering)

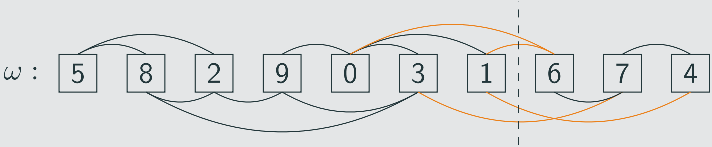


$\mathrm{cw}(G, \omega) = \max\left(\{2, 3, 2, 3, 5, 3,\right.$

Cutwidth

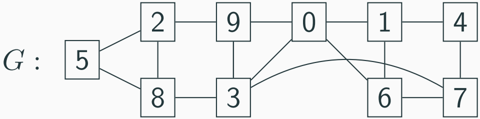


Definition (Cutwidth of an ordering)

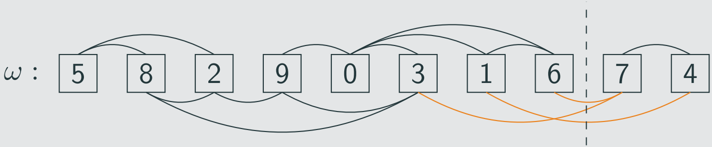


$\text{cw}(G, \omega) = \max\left(\{2, 3, 2, 3, 5, 3, 4,\right.$

Cutwidth

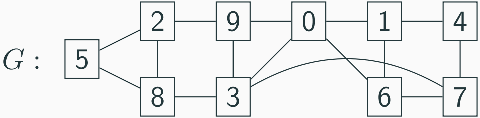


Definition (Cutwidth of an ordering)

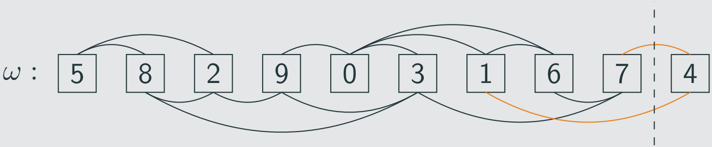


$\mathrm{cw}(G, \omega) = \max\left(\{2, 3, 2, 3, 5, 3, 4, 3,\right.$

Cutwidth

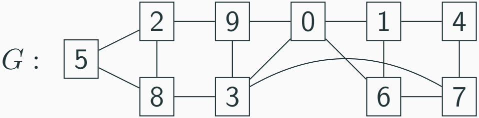


Definition (Cutwidth of an ordering)

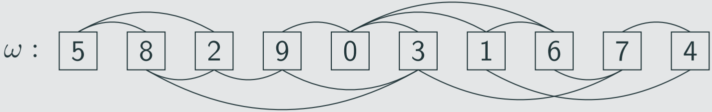


$\mathrm{cw}(G, \omega) = \max(\{2, 3, 2, 3, 5, 3, 4, 3, 2\}) = 5$

Cutwidth



Definition (Cutwidth of an ordering)



$\mathrm{cw}(G, \omega) = \max\left(\{2, 3, 2, 3, 5, 3, 4, 3, 2\}\right) = 5$

Definition (Cutwidth)

$\mathrm{cw}(G) = \min_{\omega}\left(\mathrm{cw}(G, \omega)\right)$

Order Reconfiguration

Definition (Swap)

5	8	2	9	0	3	1	6	7	4

Order Reconfiguration

Definition (Swap)

5 8 2 9 0 **3** **1** 6 7 4

Order Reconfiguration

Definition (Swap)

Order Reconfiguration

Definition (Swap)

5	8	2	9	0	1	3	6	7	4

Definition (Order reconfiguration)

ω can be reconfigured into ω' if

$$\omega = \omega_0 \to \omega_1 \to \cdots \to \omega_r = \omega'.$$

Order Reconfiguration

Definition (Swap)

5 8 2 9 0 1 3 6 7 4

Definition (Order reconfiguration)

ω can be reconfigured into ω' if

$$\omega = \omega_0 \to \omega_1 \to \cdots \to \omega_r = \omega'.$$

Problem (Bounded Cutwidth Order Reconfiguration)

Let G be an n-vertex graph, $\omega, \omega' : [n] \to V(G)$ be linear orders on the vertex set of G, and $k \in \mathbb{N}$. Is it true that ω can be reconfigured into ω' in cutwidth at most k?

Bounded Cutwidth Order Reconfiguration

Theorem

Let G be a graph and ω, ω' be linear orders of $V(G)$ of cutwidth at most k. Then, ω can be reconfigured into ω' in cutwidth at most $\mathrm{cw}(G, \omega) + \mathrm{cw}(G, \omega') \leq 2k$.

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

ω :	5	8	2	9	0	3	1	6	7	4

Proof: Big Steps

ω : 5 8 2 9 0 3 1 6 7 4

ω' : 0 1 2 3 4 5 6 7 8 9

ω : 5 8 2 9 0 3 1 6 7 4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_1 \omega$:	0	5	8	2	9	3	1	6	7	4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_1 \omega$:	0	5	8	2	9	3	1	6	7	4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_1 \omega$:	0	5	8	2	9	3	1	6	7	4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_2 \omega$:	0	1	5	8	2	9	3	6	7	4

Proof: Big Steps

ω : 5 8 2 9 0 3 1 6 7 4

ω' : 0 1 2 3 4 5 6 7 8 9

$\omega' \oplus_2 \omega$: 0 1 5 8 2 9 3 6 7 4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_3 \omega$:	0	1	2	5	8	9	3	6	7	4

Proof: Big Steps

ω : 5 8 2 9 0 3 1 6 7 4

ω' : 0 1 2 3 4 5 6 7 8 9

$\omega' \oplus_3 \omega$: 0 1 2 5 8 9 3 6 7 4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_6 \omega$:	0	1	2	3	4	5	8	9	6	7

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_7 \omega$:	0	1	2	3	4	5	6	8	9	7

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_8 \omega$:	0	1	2	3	4	5	6	7	8	9

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_9 \omega$:	0	1	2	3	4	5	6	7	8	9

Proof: Big Steps

ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

ω' :	0	1	2	3	4	5	6	7	8	9

Proof: Big Steps

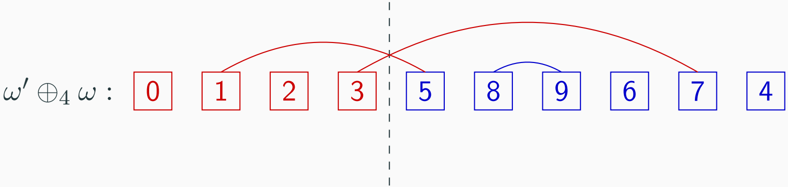
ω :	5	8	2	9	0	3	1	6	7	4
ω' :	0	1	2	3	4	5	6	7	8	9

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4

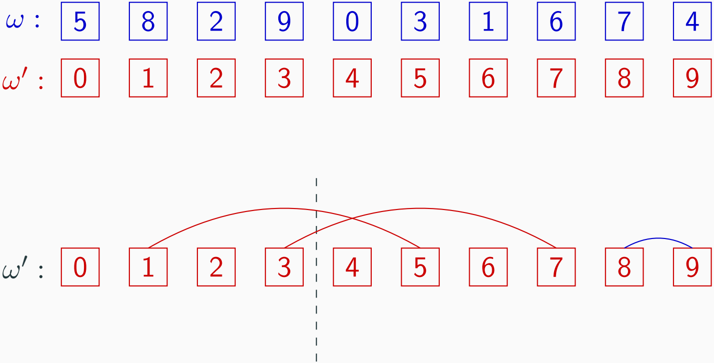
Proof: Big Steps

ω : 5 8 2 9 0 3 1 6 7 4

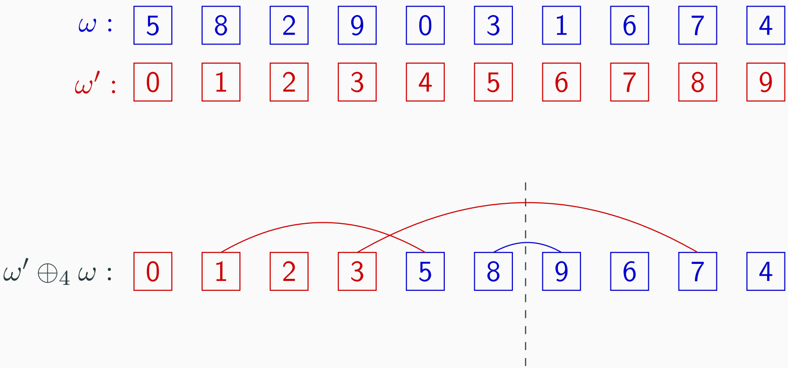
ω' : 0 1 2 3 4 5 6 7 8 9



Proof: Big Steps



Proof: Big Steps



Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

0	1	2	3	5	8	9	6	7	4

Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

0	1	2	3	5	8	9	6	4	7

Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

0	1	2	3	5	8	9	4	6	7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

0 1 2 3 5 8 4 9 6 7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

0 1 2 3 5 4 8 9 6 7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

0 1 2 3 4 5 8 9 6 7

Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

0	1	2	3	5	8	9	4	6	7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

0 1 2 3 5 8 9 | 4 6 7

Proof: Small Steps

$\omega' \oplus_4 \omega$:	0	1	2	3	5	8	9	6	7	4
$\omega' \oplus_5 \omega$:	0	1	2	3	4	5	8	9	6	7

0	1	2	3	5	8	9	4	6	7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

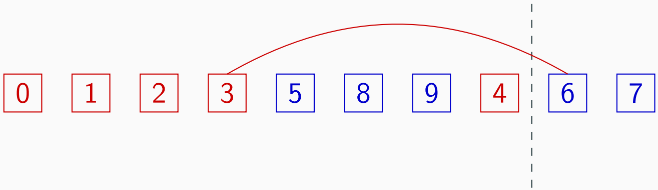
$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

0 1 2 3 5 8 9 4 | 6 7

Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

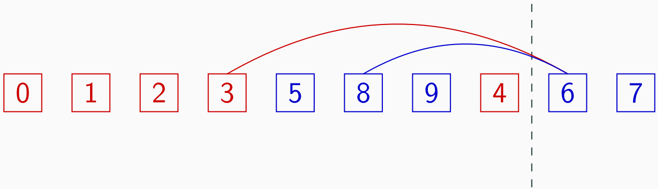
$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7



Proof: Small Steps

$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

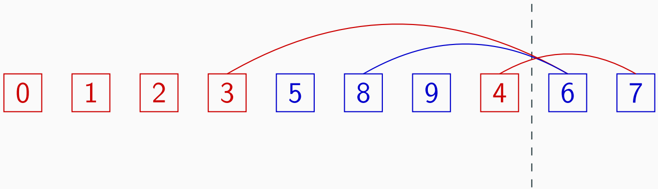
$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7



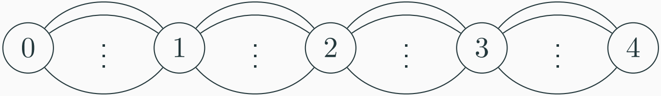
Proof: Small Steps

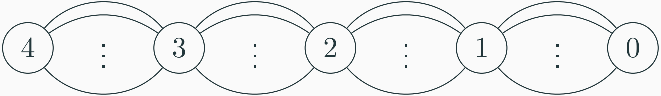
$\omega' \oplus_4 \omega$: 0 1 2 3 5 8 9 6 7 4

$\omega' \oplus_5 \omega$: 0 1 2 3 4 5 8 9 6 7

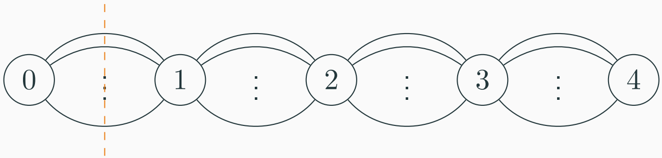


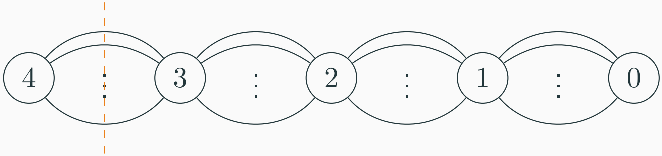
Lower Bound Construction



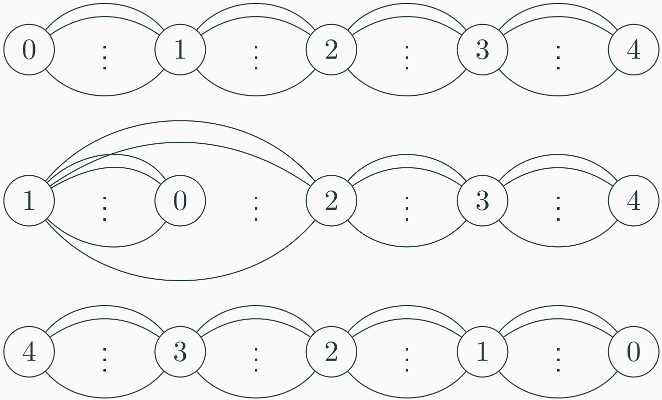


Lower Bound Construction

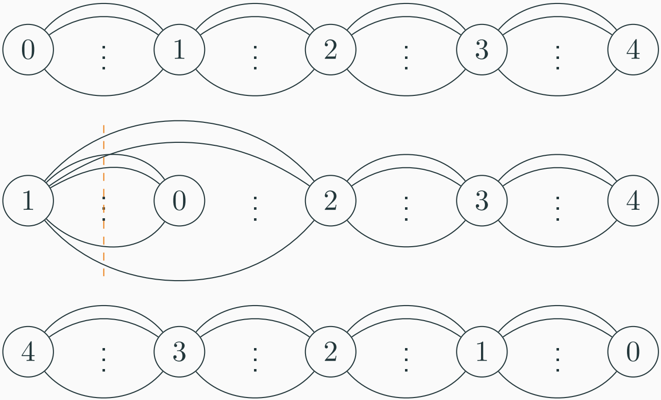




Lower Bound Construction

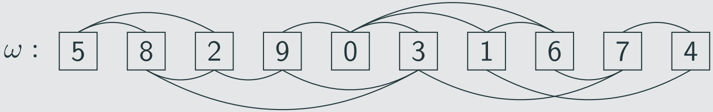


Lower Bound Construction



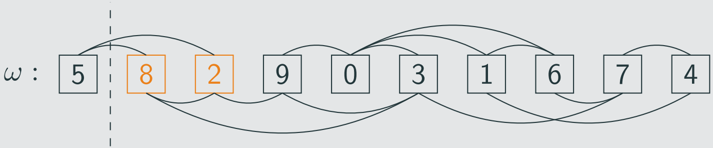
Bounded Vertex Separation Number Order Reconfiguration

Definition (Vertex separation number)



Bounded Vertex Separation Number Order Reconfiguration

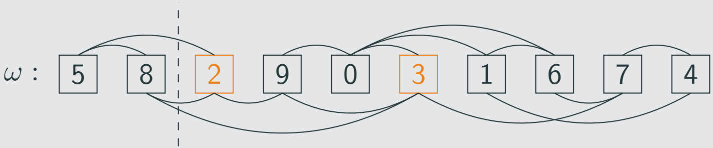
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2,$

Bounded Vertex Separation Number Order Reconfiguration

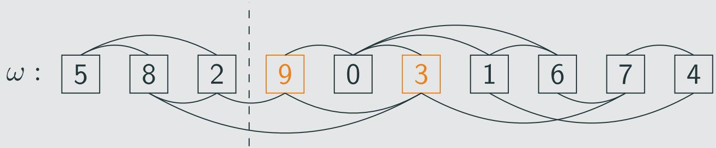
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2, 2,$

Bounded Vertex Separation Number Order Reconfiguration

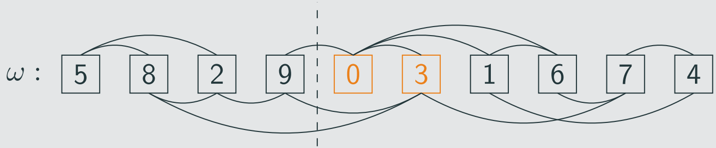
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max\left(\{2, 2, 2,\right.$

Bounded Vertex Separation Number Order Reconfiguration

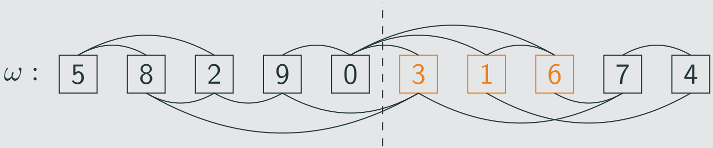
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2, 2, 2, 2,$

Bounded Vertex Separation Number Order Reconfiguration

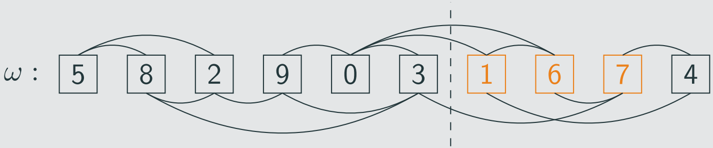
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max\left(\{2, 2, 2, 2, 3,\right.$

Bounded Vertex Separation Number Order Reconfiguration

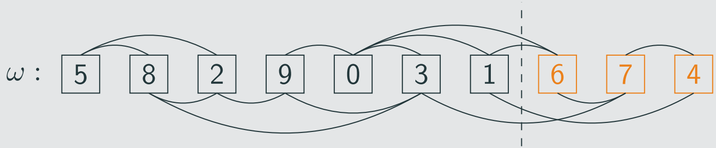
Definition (Vertex separation number)



$\text{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3,$

Bounded Vertex Separation Number Order Reconfiguration

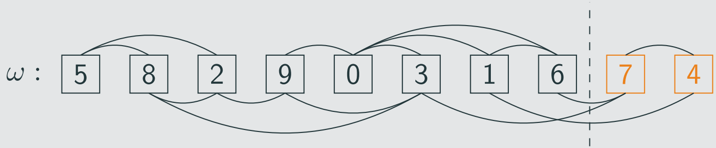
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3,$

Bounded Vertex Separation Number Order Reconfiguration

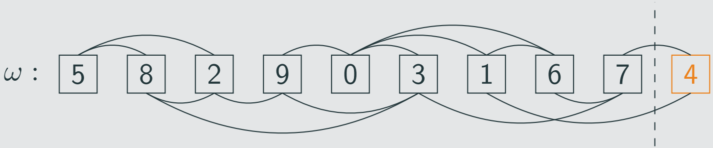
Definition (Vertex separation number)



$\text{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3, 2,$

Bounded Vertex Separation Number Order Reconfiguration

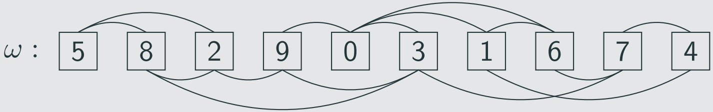
Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3, 2, 1\}) = 3$

Bounded Vertex Separation Number Order Reconfiguration

Definition (Vertex separation number)

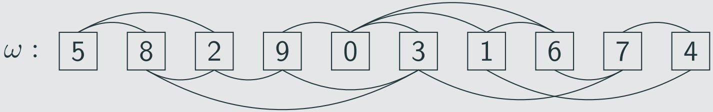


$\text{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3, 2, 1\}) = 3$

$\text{vsn}(G) = \min_{\omega}(\text{vsn}(G, \omega))$

Bounded Vertex Separation Number Order Reconfiguration

Definition (Vertex separation number)



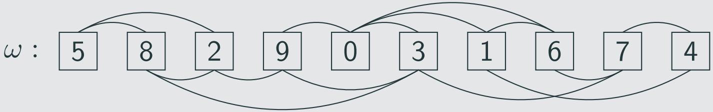
$\mathrm{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3, 2, 1\}) = 3$

$\mathrm{vsn}(G) = \min_{\omega}(\mathrm{vsn}(G, \omega))$

Known (Kinnersley IPL 1992) $\mathrm{vsn}(G) = \mathrm{pw}(G)$

Bounded Vertex Separation Number Order Reconfiguration

Definition (Vertex separation number)



$\mathrm{vsn}(G, \omega) = \max(\{2, 2, 2, 2, 3, 3, 3, 2, 1\}) = 3$

$\mathrm{vsn}(G) = \min_\omega (\mathrm{vsn}(G, \omega))$

Known (Kinnersley IPL 1992) $\mathrm{vsn}(G) = \mathrm{pw}(G)$

Theorem

Let G be a graph and ω, ω' be linear orders of $V(G)$ of vertex separation number at most k. Then, ω can be reconfigured into ω' in vertex separation number at most $\mathrm{vsn}(G, \omega) + \mathrm{vsn}(G, \omega') \leq 2k$.

Slice rewriting system

String Rewriting System

Definition (String Rewriting System)

A string rewriting system is a pair (Σ, R) where Σ is a finite alphabet, and $R \subseteq \Sigma^* \times \Sigma^*$ is a set of rewriting rules.

String Rewriting System

Definition (String Rewriting System)

A string rewriting system is a pair (Σ, R) where Σ is a finite alphabet, and $R \subseteq \Sigma^* \times \Sigma^*$ is a set of rewriting rules.

Example

With the rule $ab \to cd$, we can rewrite $abba$ into $cdba$.

String Rewriting System

Definition (String Rewriting System)

A string rewriting system is a pair (Σ, R) where Σ is a finite alphabet, and $R \subseteq \Sigma^* \times \Sigma^*$ is a set of rewriting rules.

Example

With the rule $ab \to cd$, we can rewrite $abba$ into $cdba$.

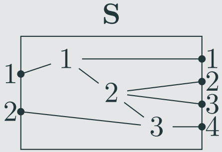
Problem (Reachability)

Given two strings w and w' in Σ^*, is there a sequence of rewrites that transforms w into w'?

Slices

Definition (Slice)

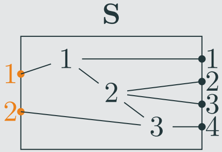
A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



Slices

Definition (Slice)

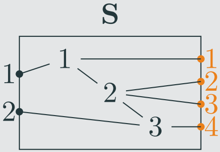
A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



Slices

Definition (Slice)

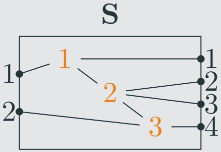
A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



Slices

Definition (Slice)

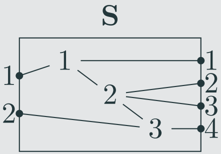
A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



Slices

Definition (Slice)

A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.

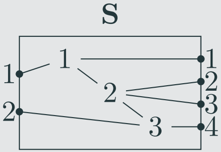


The width of $\mathbf{S}$ is $w(\mathbf{S}) = \max\{|I|, |O|\}$

Slices

Definition (Slice)

A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



The width of $\mathbf{S}$ is $w(\mathbf{S}) = \max\{|I|, |O|\}$

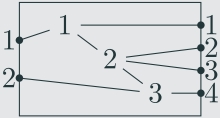
Definition (Unit Slice)

Slices

Definition (Slice)

A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.

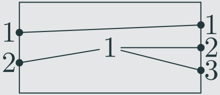
$\mathbf{S}$



The width of $\mathbf{S}$ is $w(\mathbf{S}) = \max\{|I|, |O|\}$

Definition (Unit Slice)

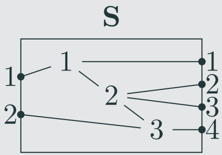
$\mathbf{S}_u$



Slices

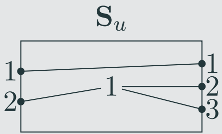
Definition (Slice)

A slice is a (multi-)graph $G = (V, E)$ such that $V = I \cup C \cup O$.



The width of $\mathbf{S}$ is $w(\mathbf{S}) = \max\{|I|, |O|\}$

Definition (Unit Slice)

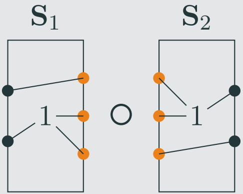


For each $k \in \mathbb{N}$, we define the alphabet $\mathbf{\Sigma}(k)$ as the set of all unit slices of width at most k.

Definition (Gluing)

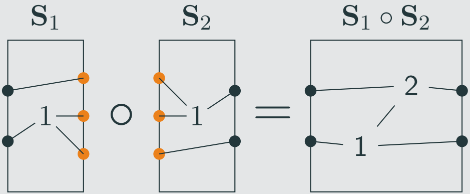
Gluing of Slices

Definition (Gluing)



Gluing of Slices

Definition (Gluing)

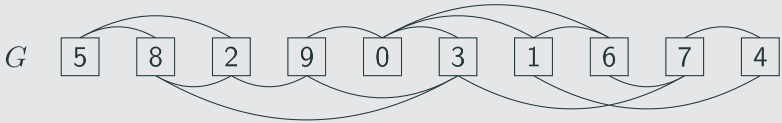


Unit Decompositions

Definition (Unit Decomposition)

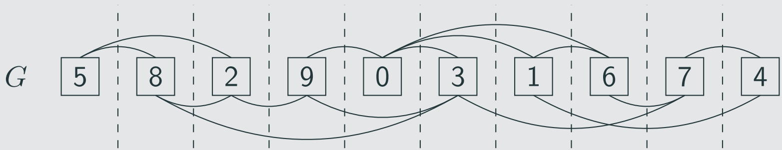
Unit Decompositions

Definition (Unit Decomposition)



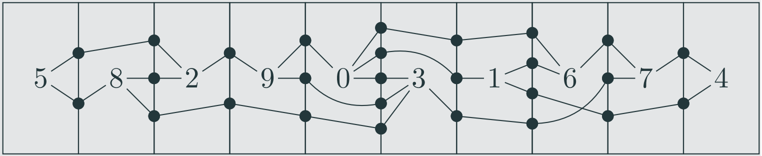
Unit Decompositions

Definition (Unit Decomposition)



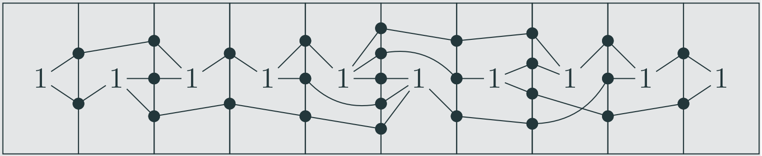
Unit Decompositions

Definition (Unit Decomposition)



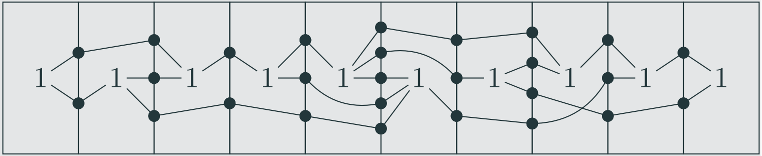
Definition (Unit Decomposition)

$\mathbf{U}_G$



Unit Decompositions

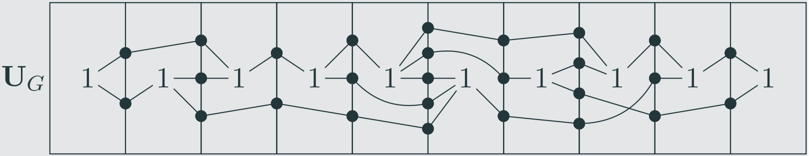
Definition (Unit Decomposition)

$\mathbf{U}_G$ 

- The gluing $\mathring{\mathbf{U}}_G$ of a unit decomposition $\mathbf{U}_G$ of a graph G is isomorphic to G.

Unit Decompositions

Definition (Unit Decomposition)



- The gluing $\mathring{\mathbf{U}}_G$ of a unit decomposition $\mathbf{U}_G$ of a graph G is isomorphic to G.
- $\mathbf{U}_G$ defines a linear order $\omega_{\mathbf{U}_G}$ of $V(\mathring{\mathbf{U}}_G)$.

Slice Equivalence Relation

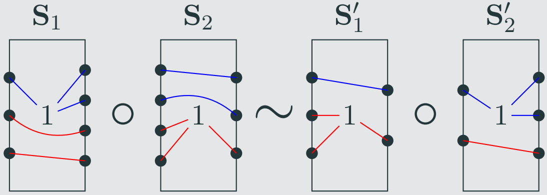
Definition (Equivalence of unit slices)

$\mathbf{S}_1\mathbf{S}_2 \sim \mathbf{S}'_1\mathbf{S}'_2$ iff there exist an isomorphism φ from $\mathbf{S}_1 \circ \mathbf{S}_2$ to $\mathbf{S}'_1 \circ \mathbf{S}'_2$

Slice Equivalence Relation

Definition (Equivalence of unit slices)

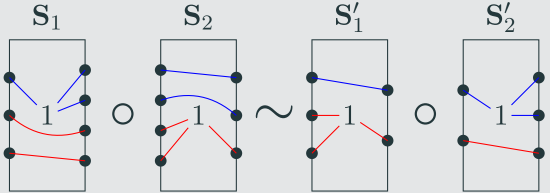
$\mathbf{S}_1\mathbf{S}_2 \sim \mathbf{S}'_1\mathbf{S}'_2$ iff there exist an isomorphism φ from $\mathbf{S}_1 \circ \mathbf{S}_2$ to $\mathbf{S}'_1 \circ \mathbf{S}'_2$



Slice Equivalence Relation

Definition (Equivalence of unit slices)

$\mathbf{S}_1\mathbf{S}_2 \sim \mathbf{S}_1'\mathbf{S}_2'$ iff there exist an isomorphism φ from $\mathbf{S}_1 \circ \mathbf{S}_2$ to $\mathbf{S}_1' \circ \mathbf{S}_2'$

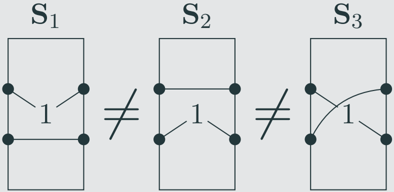


Definition (Slice rewriting system)

$\mathcal{R}(k) = \{\mathbf{S}_1\mathbf{S}_2 \to \mathbf{S}_1'\mathbf{S}_2' \ : \ \mathbf{S}_1\mathbf{S}_2 \sim \mathbf{S}_1'\mathbf{S}_2'\} \subseteq \mathbf{\Sigma}(k)^2 \times \mathbf{\Sigma}(k)^2$

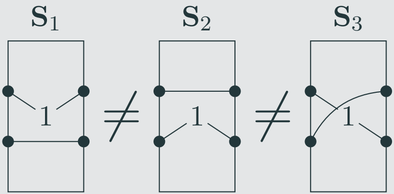
Slice Equality and Twisting

Equality

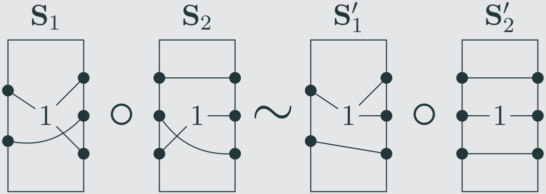


Slice Equality and Twisting

Equality



Twisting



Graph Isomorphism and Reachability

Theorem

Let $\mathbf{U}$ *and* $\mathbf{U}'$ *be unit decompositions in* $\mathbf{\Sigma}(k)^{\circledast}$*. Then,* $\mathring{\mathbf{U}}$ *is isomorphic to* $\mathring{\mathbf{U}}'$ *if and only if* $\mathbf{U}'$ *is reachable from* $\mathbf{U}$ *using* $\mathcal{R}(2k)$*.*

Graph Isomorphism and Reachability

Theorem

Let $\mathbf{U}$ and $\mathbf{U}'$ be unit decompositions in $\boldsymbol{\Sigma}(k)^{\circledast}$. Then, $\mathring{\mathbf{U}}$ is isomorphic to $\mathring{\mathbf{U}}'$ if and only if $\mathbf{U}'$ is reachable from $\mathbf{U}$ using $\mathcal{R}(2k)$.

Theorem (Giannopoulou et al. Algorithmica 2019)

Let G be an n-vertex graph of cutwidth k. We can compute a linear order ω of the vertices of G of width k in time $k^{\mathcal{O}(k^2)} \cdot n$.

Theorem

Graph isomorphism for n-vertex graphs of cutwidth at most k can be reduced in time $k^{\mathcal{O}(k^2)} \cdot n$ to $\mathcal{R}(2k)$-reachability.

Reconfiguring Orders in General

The General Picture

- Reconfiguration problems where solutions are given by orderings are an interesting area.

The General Picture

- Reconfiguration problems where solutions are given by orderings are an interesting area.
- Many concrete 'classical reconfiguration questions' are open / untouched, e.g.: What is the complexity of the next problem: Given two orderings ω, ω' of cutwidth $\leq k$ of a graph G and an integer $\ell \geq k$, is it possible to reconfigure ω into ω', so that all intermediate orderings have cutwidth $\leq \ell$?

The General Picture

- Reconfiguration problems where solutions are given by orderings are an interesting area.
- Many concrete 'classical reconfiguration questions' are open / untouched, e.g.: What is the complexity of the next problem: Given two orderings ω, ω' of cutwidth $\leq k$ of a graph G and an integer $\ell \geq k$, is it possible to reconfigure ω into ω', so that all intermediate orderings have cutwidth $\leq \ell$?
- One can find may more 'ordering questions' on graphs and strings that lead to 'swap' as a basic operation and where similar reconfiguration problems can be formulated.

The General Picture

- Reconfiguration problems where solutions are given by orderings are an interesting area.
- Many concrete 'classical reconfiguration questions' are open / untouched, e.g.: What is the complexity of the next problem: Given two orderings ω, ω' of cutwidth $\leq k$ of a graph G and an integer $\ell \geq k$, is it possible to reconfigure ω into ω', so that all intermediate orderings have cutwidth $\leq \ell$?
- One can find may more 'ordering questions' on graphs and strings that lead to 'swap' as a basic operation and where similar reconfiguration problems can be formulated.
- These have also practical 'dynamic aspects', as explained with two examples next.

One Side Crossing Minimization (OSCM)

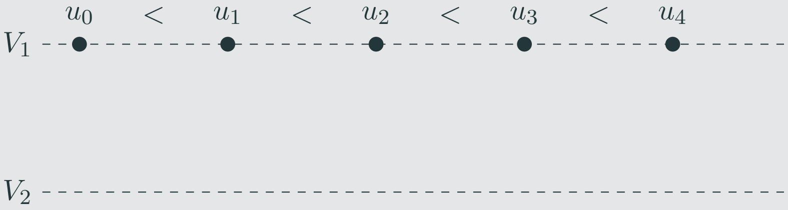
Definition (Two-layer drawing)

Let $G = (V_1, V_2, E)$ be a **bipartite** graph.

One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

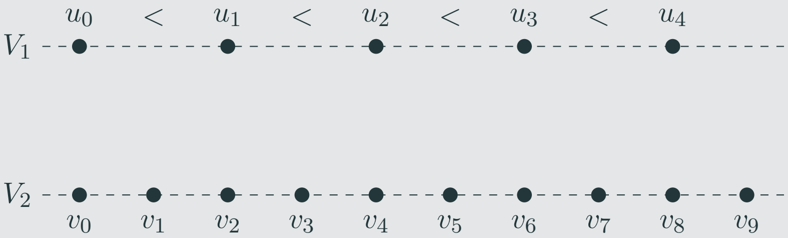
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

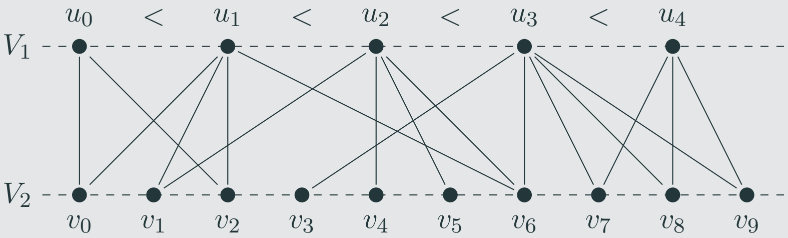
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

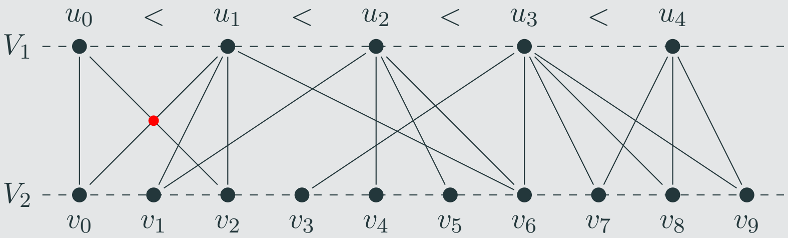
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

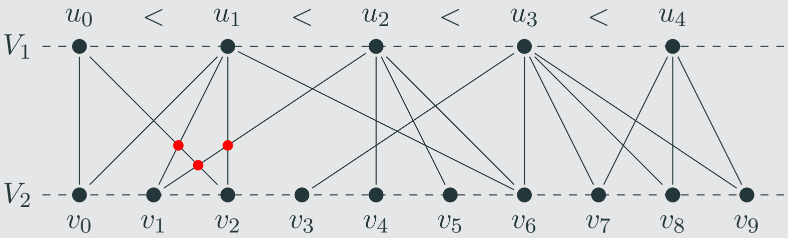
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

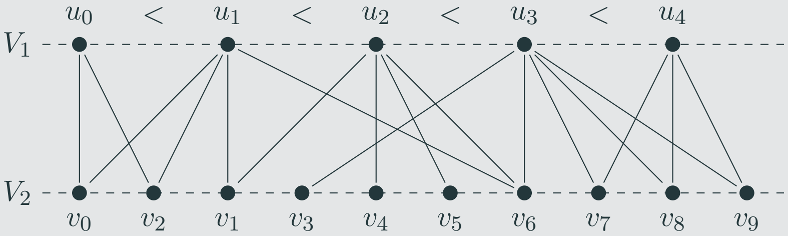
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

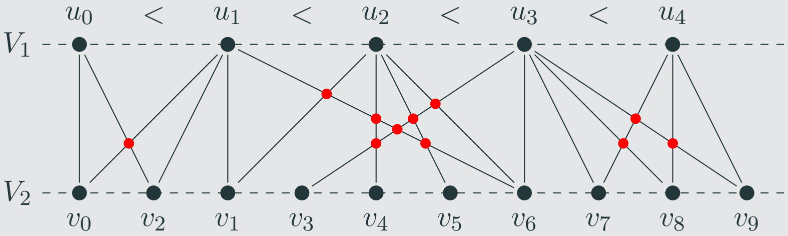
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

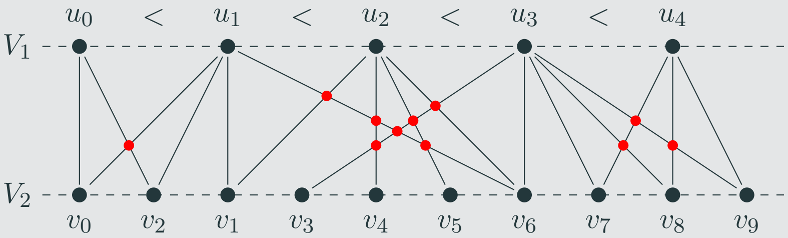
Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



One Side Crossing Minimization (OSCM)

Definition (Two-layer drawing)

Let $G = (V_1, V_2, E)$ be a **bipartite** graph.



Problem (OSCM)

*Given a **bipartite graph** $G = (V_1, V_2, E)$, a **linear order** τ_1 on V_1 and $k \in \mathbb{N}$. Is there a **linear order** τ_2 on V_2 such that the two-layer drawing specified by (τ_1, τ_2) has at most k edge crossings?*

Grouping by Swapping (GbS)

Definition (Swap)

c	a	d	d	a	a	b	c	c	d	d

Grouping by Swapping (GbS)

Definition (Swap)

c	a	d	d	a	a	b	c	c	d	d

Grouping by Swapping (GbS)

Definition (Swap)

c a d d a b ↔ a c c d d

Grouping by Swapping (GbS)

Definition (Swap)

c a d d a b a c c d d

Grouping by Swapping (GbS)

Definition (Swap)

c	a	d	d	a	b	a	c	c	d	d

Definition (Block string)

Grouping by Swapping (GbS)

Definition (Swap)

c a d d a b a c c d d

Definition (Block string)

a a a c c c b d d d d

Grouping by Swapping (GbS)

Definition (Swap)

c a d d a b a c c d d

Definition (Block string)

a a a	c c c	b	d d d d

Grouping by Swapping (GbS)

Definition (Swap)

c a d d a b a c c d d

Definition (Block string)

| a a a | c c c | b | d d d d |

Problem (GbS)

*Given a **finite alphabet** Σ, a **string** $w \in \Sigma^*$, and $k \in \mathbb{N}$. Can we transform w in a **block string** w' with at most k **swaps**?*

Reduction from GbS to OSCM

GbS

An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.

Reduction from GbS to OSCM

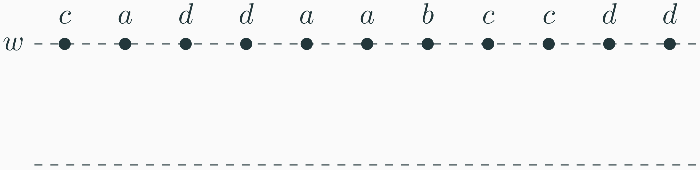
GbS

An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.

Reduction from GbS to OSCM

GbS

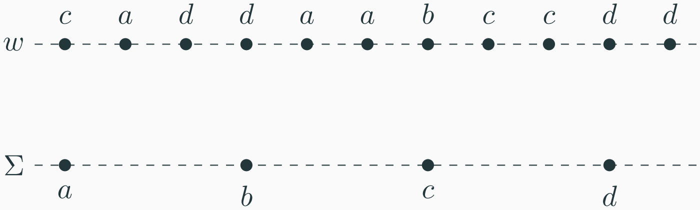
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

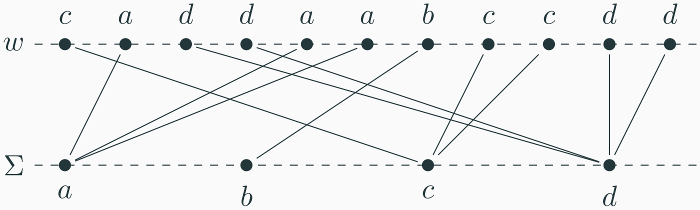
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

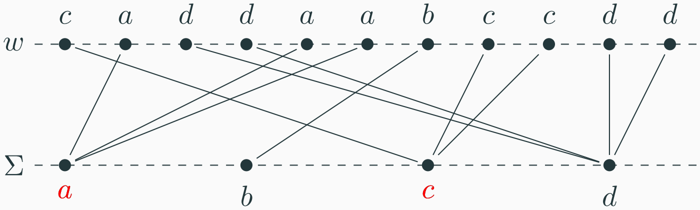
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

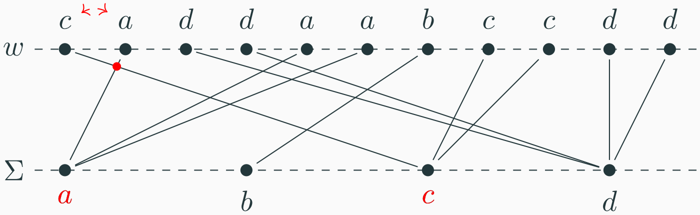
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

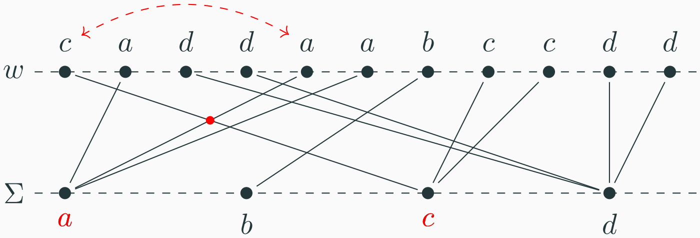
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

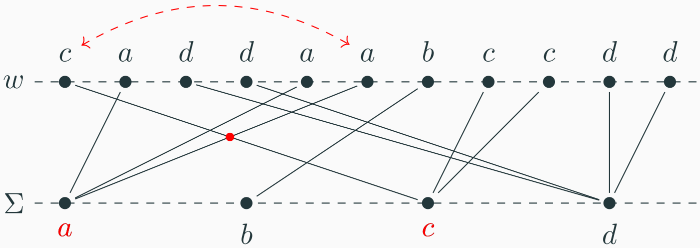
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

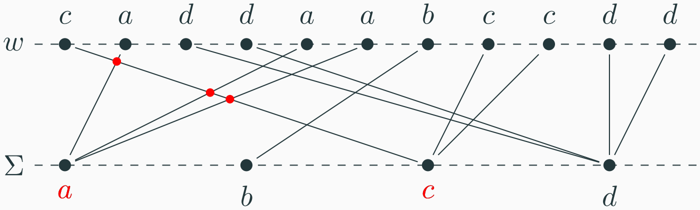
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

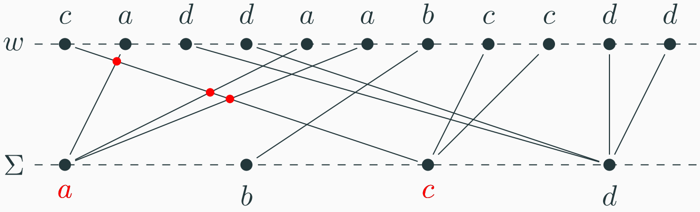
An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



Reduction from GbS to OSCM

GbS

An alphabet $\Sigma = \{a, b, c, d\}$ and $w = caddaabccdd$.



See E. Arrighi et al. FSTTCS 2020 & IJCAI 2021.

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.
- Bubble sort can be helpful, seemingly providing a quadratic upper-bound on the length of reconfiguration sequences.

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.
- Bubble sort can be helpful, seemingly providing a quadratic upper-bound on the length of reconfiguration sequences.
- But: if additional parameters (like cutwidth) are limited, then possibly ‘detours’ are necessary, or no way exists at all!

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.
- Bubble sort can be helpful, seemingly providing a quadratic upper-bound on the length of reconfiguration sequences.
- But: if additional parameters (like cutwidth) are limited, then possibly ‘detours’ are necessary, or no way exists at all!
- Again, computational complexity questions unexplored!

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.
- Bubble sort can be helpful, seemingly providing a quadratic upper-bound on the length of reconfiguration sequences.
- But: if additional parameters (like cutwidth) are limited, then possibly ‘detours’ are necessary, or no way exists at all!
- Again, computational complexity questions unexplored!
- Also, no understanding of the structure of the solution space.

Homework for Reconfigurationalists

- Explore the search space of orderings, with neighborhood defined by a single swap.
- Bubble sort can be helpful, seemingly providing a quadratic upper-bound on the length of reconfiguration sequences.
- But: if additional parameters (like cutwidth) are limited, then possibly 'detours' are necessary, or no way exists at all!
- Again, computational complexity questions unexplored!
- Also, no understanding of the structure of the solution space.
- Conversely: we explained connections to string rewriting. Can we make use of other rewriting theory results in reconfiguration?

Thank you!

Trier, Germany
June 7th-9th

IWOCA 2022