# Graph and String Parameters: Connections Between Pathwidth, Cutwidth and the Locality Number

Katrin Casel[1], Joel D. Day[2], Pamela Fleischmann[3], Tomasz Kociumaka[4], Florin Manea[3], **Markus L. Schmid**[5]

[1] HPI, University of Potsdam, Germany
[2] Loughborough University, UK
[3] Kiel University, Germany
[4] University of Warsaw, Poland, and Bar-Ilan University, Israel
[5] Trier University, Germany

Theorietag 2019 – Marburg

# A Solitaire Game on Strings

### The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence:

marked blocks:
maximum number of marked blocks:

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

`a d a b a d b d a e c b c b`

marking sequence: `b`

marked blocks: 4
maximum number of marked blocks: 4

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence: b, c

marked blocks: 3
maximum number of marked blocks: 4

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence: b, c, e

marked blocks: 3
maximum number of marked blocks: 4

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence: b, c, e, d

marked blocks: 4

maximum number of marked blocks: 4

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence: b, c, e, d, a

marked blocks: 1
maximum number of marked blocks: 4

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence:

marked blocks:
maximum number of marked blocks:

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

`a d a b a d b d a e c b c b`

marking sequence: `d`

marked blocks: 3
maximum number of marked blocks: 3

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

a d a b a d b d a e c b c b

marking sequence: d, a

marked blocks: 3
maximum number of marked blocks: 3

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

`a d a b a d b d a e c b c b`

marking sequence: `d`, `a`, `b`

marked blocks: 3
maximum number of marked blocks: 3

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

`a d a b a d b d a e c b c b`

marking sequence: `d`, `a`, `b`, `c`

marked blocks: 2
maximum number of marked blocks: 3

# A Solitaire Game on Strings

## The Game

Given: String $\alpha$ over (finite) alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

Objective: Mark all symbols $a_1, a_2, \ldots, a_n$ in some order (all occ. of the same symbol in parallel), such that there are only few contiguous blocks of marked symbols in the word.

## Example

`a d a b a d b d a e c b c b`

marking sequence: `d`, `a`, `b`, `c`, `e`

marked blocks: 1
maximum number of marked blocks: 3

# The Locality Number

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite alphabet.

## Marking Sequence

Any ordered list $\sigma$ of the symbols from $X$ (or, equivalently, a bijection $\sigma : \{1, 2, \ldots, |X|\} \to X$) is a **marking sequence**.

# The Locality Number

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite alphabet.

## Marking Sequence

Any ordered list $\sigma$ of the symbols from $X$ (or, equivalently, a bijection $\sigma : \{1, 2, \ldots, |X|\} \to X$) is a **marking sequence**.

## Marking Number

The **marking number** $\pi_\sigma(\alpha)$ (of $\sigma$ with respect to $\alpha$) is the maximum number of marked blocks obtained while marking $\alpha$ according to $\sigma$.

# The Locality Number

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite alphabet.

## Marking Sequence

Any ordered list $\sigma$ of the symbols from $X$ (or, equivalently, a bijection $\sigma : \{1, 2, \ldots, |X|\} \to X$) is a **marking sequence**.

## Marking Number

The **marking number** $\pi_\sigma(\alpha)$ (of $\sigma$ with respect to $\alpha$) is the maximum number of marked blocks obtained while marking $\alpha$ according to $\sigma$.

## Locality Number

A string $\alpha$ over $X$ is **$k$-local** $\iff \pi_\sigma(\alpha) \leq k$, for some marking sequence $\sigma$.

The **locality number** of $\alpha$ is $\mathrm{loc}(\alpha) = \min\{k \mid \alpha \text{ is } k\text{-local}\}$.

# The Locality Number

## Example

Let $\alpha = \texttt{adabadbdaecbcb}$, $\sigma_1 = (\texttt{b}, \texttt{c}, \texttt{e}, \texttt{d}, \texttt{a})$, $\sigma_2 = (\texttt{d}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{e})$

# The Locality Number

## Example

Let $\alpha = \texttt{adabadbdaecbcb}$, $\sigma_1 = (\texttt{b}, \texttt{c}, \texttt{e}, \texttt{d}, \texttt{a})$, $\sigma_2 = (\texttt{d}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{e})$

$\pi_{\sigma_1}(\alpha) = 4$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 4$)

# The Locality Number

## Example

Let $\alpha = \texttt{adabadbdaecbcb}$, $\sigma_1 = (\texttt{b}, \texttt{c}, \texttt{e}, \texttt{d}, \texttt{a})$, $\sigma_2 = (\texttt{d}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{e})$

$\pi_{\sigma_1}(\alpha) = 4$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 4$)

$\pi_{\sigma_2}(\alpha) = 3$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 3$)

# The Locality Number

## Example

Let $\alpha = \mathtt{adabadbdaecbcb}$, $\sigma_1 = (\mathtt{b}, \mathtt{c}, \mathtt{e}, \mathtt{d}, \mathtt{a})$, $\sigma_2 = (\mathtt{d}, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{e})$

$\pi_{\sigma_1}(\alpha) = 4$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 4$)

$\pi_{\sigma_2}(\alpha) = 3$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 3$)

$\mathsf{loc}(\alpha) = 3$

# The Locality Number

## Example

Let $\alpha = \texttt{adabadbdaecbcb}$, $\sigma_1 = (\texttt{b}, \texttt{c}, \texttt{e}, \texttt{d}, \texttt{a})$, $\sigma_2 = (\texttt{d}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{e})$

$\pi_{\sigma_1}(\alpha) = 4$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 4$)

$\pi_{\sigma_2}(\alpha) = 3$ ($\Rightarrow \mathsf{loc}(\alpha) \leq 3$)

$\mathsf{loc}(\alpha) = 3$

## Motivation

Pattern matching with variables.

Marking sequence = dynamic programming algorithm

$\rightsquigarrow$ XP-algorithms w.r.t. parameter $\mathsf{loc}(\alpha)$.

# Known Results and Open Problems

## Computing the locality number

Loc

| | |
|---|---|
| Input: | String $\alpha \in \Sigma^*$, $k \in \mathbb{N}$. |
| Question: | $\mathsf{loc}(\alpha) \leq k$? |

# Known Results and Open Problems

## Computing the locality number

Loc

| | |
|---|---|
| Input: | String $\alpha \in \Sigma^*$, $k \in \mathbb{N}$. |
| Question: | $\mathsf{loc}(\alpha) \leq k$? |

MinLoc denotes the corresponding minimisation problem.

# Known Results and Open Problems

## Computing the locality number

Loc

| | |
|---|---|
| Input: | String $\alpha \in \Sigma^*$, $k \in \mathbb{N}$. |
| Question: | $\mathsf{loc}(\alpha) \leq k$? |

MinLoc denotes the corresponding minimisation problem.

## Known Results

Loc $\in$ XP w.r.t. parameter $k$ (i. e., in P for fixed $k$).

# Known Results and Open Problems

## Computing the locality number

Loc

| | |
|---|---|
| Input: | String $\alpha \in \Sigma^*$, $k \in \mathbb{N}$. |
| Question: | $\mathsf{loc}(\alpha) \leq k$? |

MinLoc denotes the corresponding minimisation problem.

## Known Results

Loc $\in$ XP w.r.t. parameter $k$ (i. e., in P for fixed $k$).

## Open Problems

- Is Loc NP-complete?
- Is Loc $\in$ FPT (w.r.t. $k$ or $|\Sigma|$)?
- Are there good approximation algorithms for MinLoc?

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \ldots, v_n\}$.

### Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \ldots, v_n\}$.

## Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

**Cut set**: $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$.

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \dots, v_n\}$.

## Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

**Cut set**: $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$.

**Size of a cut**: $|\mathcal{C}(V_1, V_2)|$.

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \ldots, v_n\}$.

## Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

**Cut set**: $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$.

**Size of a cut**: $|\mathcal{C}(V_1, V_2)|$.

## Linear Arrangements and Cutwidth

**Linear arrangement of $G$**: sequence $L = (v_{j_1}, v_{j_2}, \ldots, v_{j_n})$, where $(j_1, j_2, \ldots, j_n)$ is a permutation of $(1, 2, \ldots, n)$.

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \ldots, v_n\}$.

## Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

**Cut set**: $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$.

**Size of a cut**: $|\mathcal{C}(V_1, V_2)|$.

## Linear Arrangements and Cutwidth

**Linear arrangement of $G$**: sequence $L = (v_{j_1}, v_{j_2}, \ldots, v_{j_n})$, where $(j_1, j_2, \ldots, j_n)$ is a permutation of $(1, 2, \ldots, n)$.

**Cutwidth of $L$**:
$\mathrm{cw}(L) = \max\{|\mathcal{C}(\{v_{j_1}, v_{j_2}, \ldots, v_{j_i}\}, \{v_{j_{i+1}}, \ldots, v_{j_n}\})| \mid 0 \le i \le n\}$

# Cutwidth

Let $G = (V, E)$ be a (multi)graph with $V = \{v_1, \ldots, v_n\}$.

## Cuts

**Cut**: partition $(V_1, V_2)$ of $V$.

**Cut set**: $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$.

**Size of a cut**: $|\mathcal{C}(V_1, V_2)|$.

## Linear Arrangements and Cutwidth

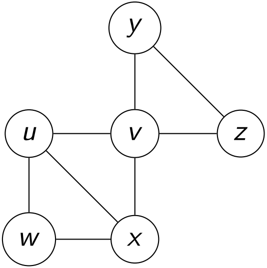
**Linear arrangement of $G$**: sequence $L = (v_{j_1}, v_{j_2}, \ldots, v_{j_n})$, where $(j_1, j_2, \ldots, j_n)$ is a permutation of $(1, 2, \ldots, n)$.

**Cutwidth of $L$**:
$\text{cw}(L) = \max\{|\mathcal{C}(\{v_{j_1}, v_{j_2}, \ldots, v_{j_i}\}, \{v_{j_{i+1}}, \ldots, v_{j_n}\})| \mid 0 \leq i \leq n\}$

**Cutwidth of $G$**: $\text{cw}(G) = \min\{\text{cw}(L) \mid L \text{ is lin. arr. for } G\}$.

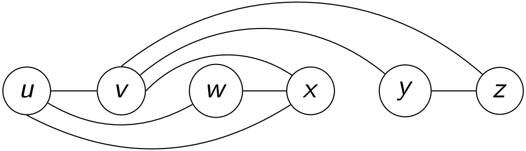
# Cutwidth – Example

Graph $G$:

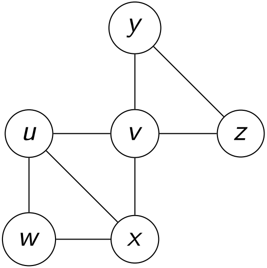


# Cutwidth – Example

Linear arrangement with cutwidth 5:

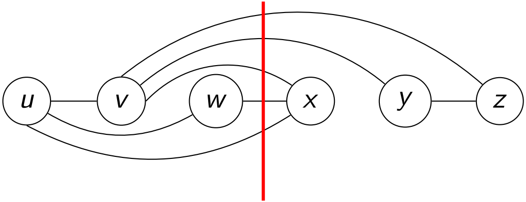



Graph $G$:

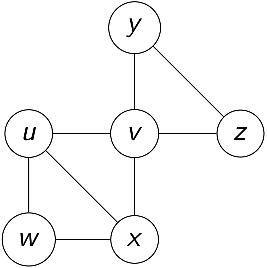


# Cutwidth – Example

Linear arrangement with cutwidth 5:

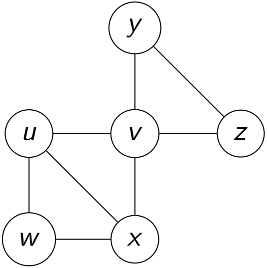



Graph $G$:

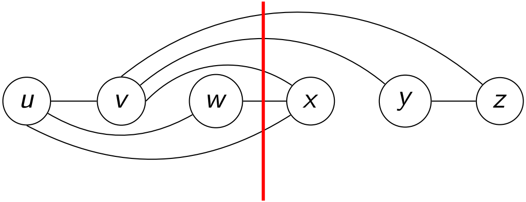


# Cutwidth – Example

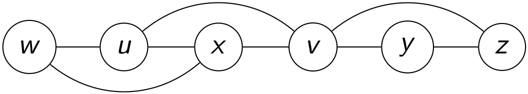
Graph $G$:




Linear arrangement with cutwidth 5:

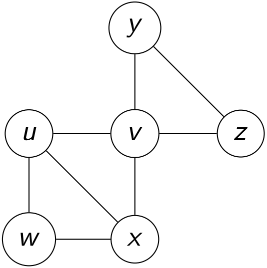



Linear arrangement with cutwidth 3:

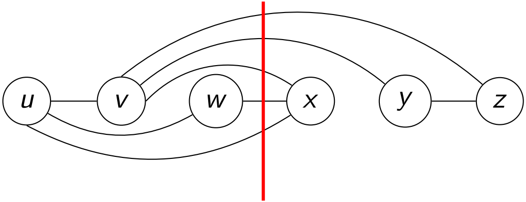


# Cutwidth – Example

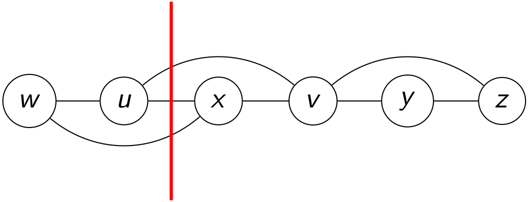
Graph $G$:




Linear arrangement with cutwidth 5:

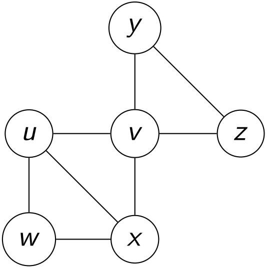



Linear arrangement with cutwidth 3:

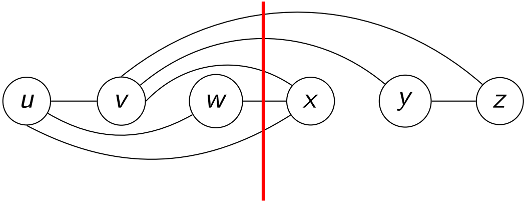


# Cutwidth – Example

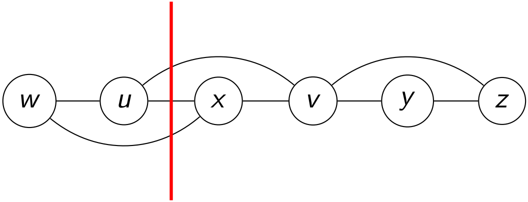
Graph $G$:




Linear arrangement with cutwidth 5:




Linear arrangement with cutwidth 3:




$\text{cw}(G) = 3$

# Computing the Cutwidth

## Cutwidth problem

Cutwidth

| | |
|---|---|
| Input: | (Multi)graph $G$, $k \in \mathbb{N}$. |
| Question: | $\mathrm{cw}(\alpha) \leq k$? |

# Computing the Cutwidth

## Cutwidth problem

Cutwidth

| | |
|---|---|
| Input: | (Multi)graph $G$, $k \in \mathbb{N}$. |
| Question: | $\mathrm{cw}(\alpha) \leq k$? |

MinCutwidth denotes the corresponding minimisation problem.

# Computing the Cutwidth

## Cutwidth problem

Cutwidth

| | |
|---|---|
| Input: | (Multi)graph $G$, $k \in \mathbb{N}$. |
| Question: | $\mathrm{cw}(\alpha) \leq k$? |

MinCutwidth denotes the corresponding minimisation problem.

## Known Results

- Cutwidth is NP-complete.
- Cutwidth $\in$ FPT (w.r.t. $k$).
- Exact exponential algorithms, linear fpt-algorithms, approximation algorithms...

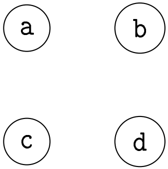
# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2$.

# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

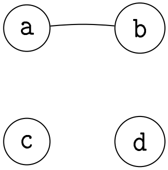
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$
$\alpha = \texttt{abcbcdbada}$
$k = 2.$

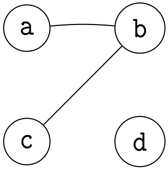
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

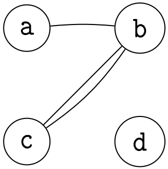
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$
$\alpha = \texttt{abcbcdbada}$
$k = 2$.

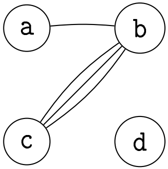
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

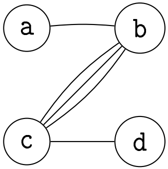
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

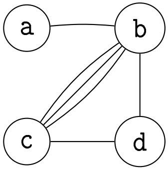
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

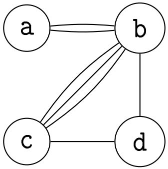
Construct multigraph
$H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

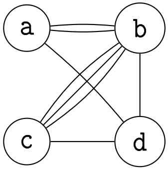
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

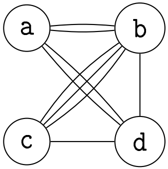
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

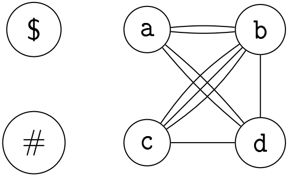
Construct multigraph
$H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

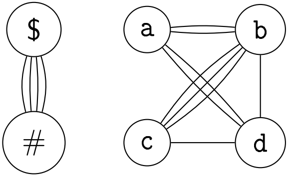
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2$.

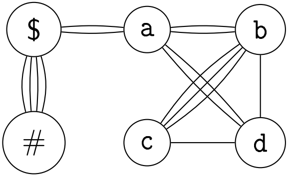
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

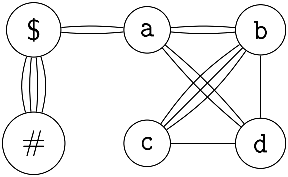
Construct multigraph $H_{\alpha,k} = (V, E)$:



# Loc $\leq$ Cutwidth

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:



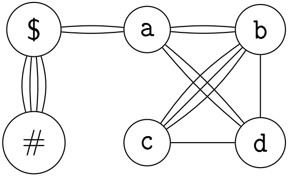

$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$

# Loc $\leq$ Cutwidth

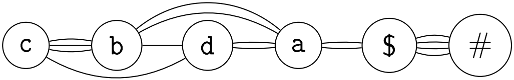
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph $H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

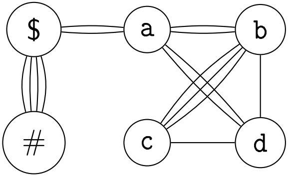
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc ≤ Cutwidth

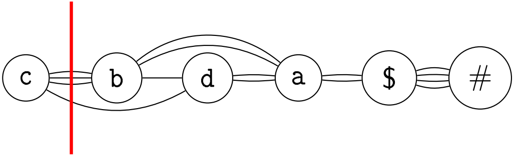
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

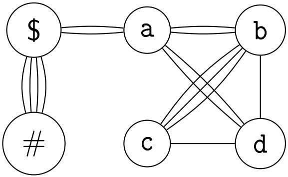
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc ≤ Cutwidth

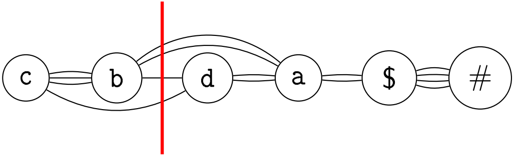
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

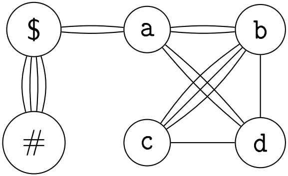
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc $\leq$ Cutwidth

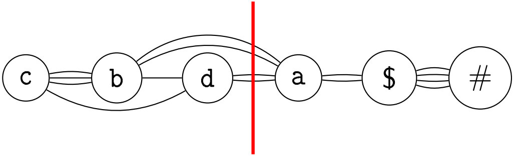
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

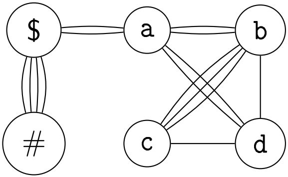
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc $\leq$ Cutwidth

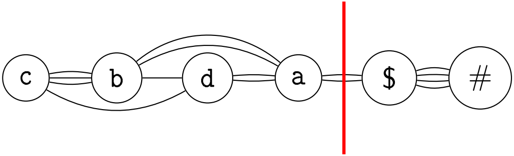
$\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$
$\alpha = \texttt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \texttt{a b c b c d b a d a}$

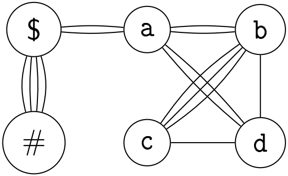
Marking sequence: $(\texttt{c}, \texttt{b}, \texttt{d}, \texttt{a})$



# Loc $\leq$ Cutwidth

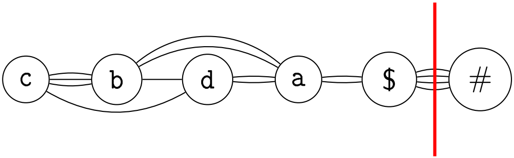
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

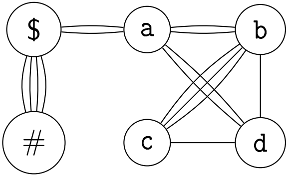
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc $\leq$ Cutwidth

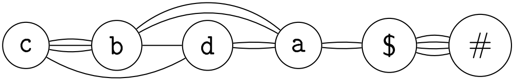
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph
$H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

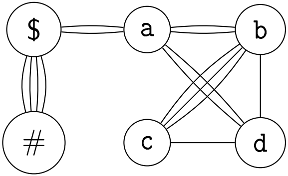
Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$



# Loc $\leq$ Cutwidth

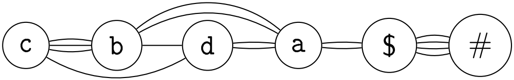
$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$\alpha = \mathtt{abcbcdbada}$
$k = 2.$

Construct multigraph $H_{\alpha,k} = (V, E)$:




$\alpha = \mathtt{a\ b\ c\ b\ c\ d\ b\ a\ d\ a}$

Marking sequence: $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$

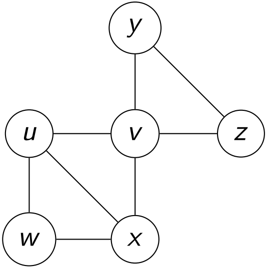



**Lemma**

$\mathrm{cw}(H_{\alpha,k}) = 2k$ if and only if $\mathrm{loc}(\alpha) \leq k$.

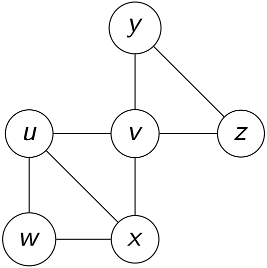
# Cutwidth $\leq$ Loc

G = (V, E):



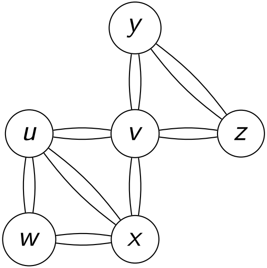
# Cutwidth $\leq$ Loc

G = (V, E):



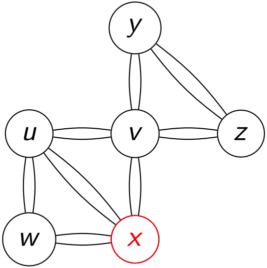
# Cutwidth $\leq$ Loc

G = (V, E):



# Cutwidth $\leq$ Loc

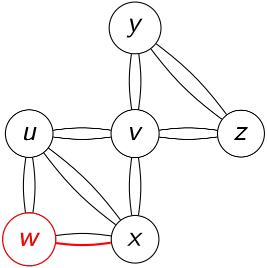
G = (V, E):




$x$

# Cutwidth ≤ Loc

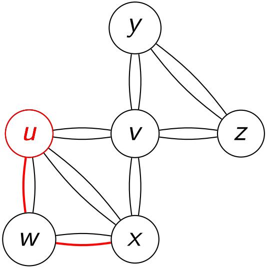
G = (V, E):




$x$ $w$

# Cutwidth ≤ Loc

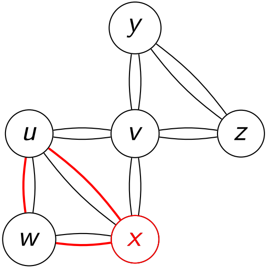
G = (V, E):




*x w* *u*

# Cutwidth $\leq$ Loc

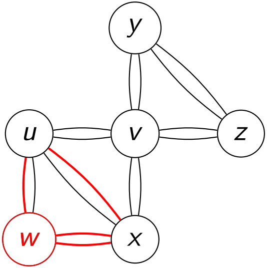
G = (V, E):




$x\ w\ u\ x$

# Cutwidth $\leq$ Loc

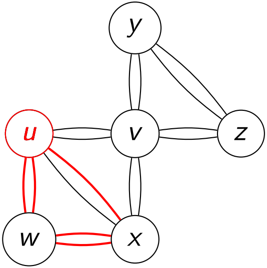
G = (V, E):




$x\ w\ u\ x\ w$

# Cutwidth $\leq$ Loc

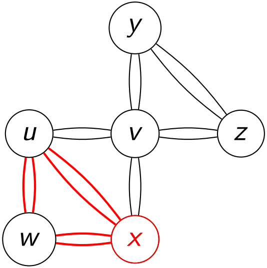
G = (V, E):




$x\ w\ u\ x\ w\ u$

# Cutwidth $\leq$ Loc

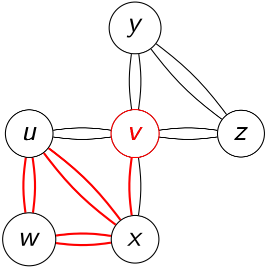
G = (V, E):




$x\ w\ u\ x\ w\ u\ x$

# Cutwidth $\leq$ Loc

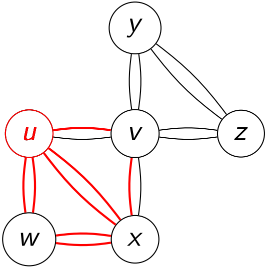
G = (V, E):




$x\ w\ u\ x\ w\ u\ x\ v$

# Cutwidth $\leq$ Loc

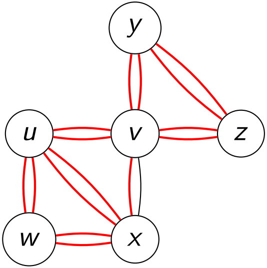
G = (V, E):




$x\ w\ u\ x\ w\ u\ x\ v\ u$

# Cutwidth $\leq$ Loc

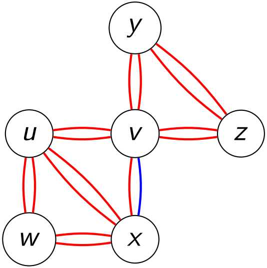
G = (V, E):




$x \; w \; u \; x \; w \; u \; x \; v \; u \; v \; y \; z \; v \; y \; z \; v$

# Cutwidth $\leq$ Loc

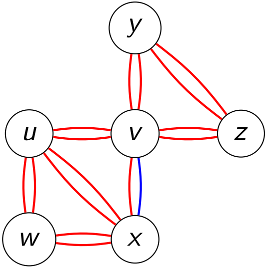
G = (V, E):




$\alpha_{\{x,v\}} = x\ w\ u\ x\ w\ u\ x\ v\ u\ v\ y\ z\ v\ y\ z\ v$

# Cutwidth $\leq$ Loc

G = (V, E):




$\alpha_{\{x,v\}} = x\ w\ u\ x\ w\ u\ x\ v\ u\ v\ y\ z\ v\ y\ z\ v$

**Lemma**

$\forall e \in E : \mathrm{cw}(G) \leq \mathrm{loc}(\alpha_e) \leq \mathrm{cw}(G) + 1$

$\exists e \in E : \mathrm{loc}(\alpha_e) = \mathrm{cw}(G).$

# Consequences

# Consequences

## Approximation Meta-Theorem

| | MinCutwidth | | MinLoc |
|---|---|---|---|
| Run time: | $O(f(\|E\|))$ | $\Rightarrow$ | $O(f(\|\alpha\|) + \|\alpha\|)$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $(r(2\,\text{opt}, \|\alpha\|) + \frac{1}{\text{opt}})$ |

# Consequences

**Approximation Meta-Theorem**

| | MinCutwidth | | MinLoc |
|---|---|---|---|
| Run time: | $O(f(\lvert E\rvert))$ | $\Rightarrow$ | $O(f(\lvert\alpha\rvert) + \lvert\alpha\rvert)$ |
| Appr. ratio: | $r(\text{opt}, \lvert E\rvert)$ | | $(r(2\,\text{opt}, \lvert\alpha\rvert) + \frac{1}{\text{opt}})$ |
| Run time: | $O(n(f(\lvert E\rvert) + \lvert E\rvert))$ | $\Leftarrow$ | $O(f(\lvert\alpha\rvert))$ |
| Appr. ratio: | $r(\text{opt}, \lvert E\rvert)$ | | $r(\text{opt}, \lvert\alpha\rvert)$ |

# Consequences

## Approximation Meta-Theorem

| | MinCutwidth | | MinLoc |
|---|---|---|---|
| Run time: | $O(f(\|E\|))$ | $\Rightarrow$ | $O(f(\|\alpha\|) + \|\alpha\|)$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $(r(2\,\text{opt}, \|\alpha\|) + \frac{1}{\text{opt}})$ |
| Run time: | $O(n(f(\|E\|) + \|E\|))$ | $\Leftarrow$ | $O(f(\|\alpha\|))$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $r(\text{opt}, \|\alpha\|)$ |

## Theorem

The problem Loc

- is NP-complete,
  (even if every symbol has at most 3 occurrences)

# Consequences

## Approximation Meta-Theorem

| | MinCutwidth | | MinLoc |
|---|---|---|---|
| Run time: | $O(f(\|E\|))$ | $\Rightarrow$ | $O(f(\|\alpha\|) + \|\alpha\|)$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $(r(2\,\text{opt}, \|\alpha\|) + \frac{1}{\text{opt}})$ |
| Run time: | $O(n(f(\|E\|) + \|E\|))$ | $\Leftarrow$ | $O(f(\|\alpha\|))$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $r(\text{opt}, \|\alpha\|)$ |

## Theorem

The problem Loc

- is NP-complete,
  (even if every symbol has at most 3 occurrences)
- can be solved in $O^*(2^{|\Sigma|})$,

# Consequences

## Approximation Meta-Theorem

| | MinCutwidth | | MinLoc |
|---|---|---|---|
| Run time: | $O(f(\|E\|))$ | $\Rightarrow$ | $O(f(\|\alpha\|) + \|\alpha\|)$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $(r(2\,\text{opt}, \|\alpha\|) + \frac{1}{\text{opt}})$ |
| Run time: | $O(n(f(\|E\|) + \|E\|))$ | $\Leftarrow$ | $O(f(\|\alpha\|))$ |
| Appr. ratio: | $r(\text{opt}, \|E\|)$ | | $r(\text{opt}, \|\alpha\|)$ |

## Theorem

The problem Loc

- is NP-complete,
  (even if every symbol has at most 3 occurrences)
- can be solved in $O^*(2^{|\Sigma|})$,
- in FPT (w.r.t. parameter $k$), with linear fpt-algorithm.

# Path-Decompositions and Pathwidth

**Path-decompositions as tree-decomposition**

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

- Initially all vertices are blue.

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

- Initially all vertices are blue.
- Until all vertices are blue again,
  - color a vertex red that has never been red before, or
  - color a red vertex blue again,

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

- Initially all vertices are blue.
- Until all vertices are blue again,
  - color a vertex red that has never been red before, or
  - color a red vertex blue again,
- such that each two adjacent vertices are red at the same time.

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

- Initially all vertices are blue.
- Until all vertices are blue again,
  - color a vertex red that has never been red before, or
  - color a red vertex blue again,
- such that each two adjacent vertices are red at the same time.

**pw**$(Q)$: Max. number of marked vertices.

# Path-Decompositions and Pathwidth

## Path-decompositions as tree-decomposition

A path-decomposition is a tree-decomposition the underlying tree-structure of which is a path.

## Path-decompositions as marking procedures

Let $G = (V, E)$ be a graph.

**Path-decomposition** $Q$ of $G$:

- Initially all vertices are blue.
- Until all vertices are blue again,
  - color a vertex red that has never been red before, or
  - color a red vertex blue again,
- such that each two adjacent vertices are red at the same time.

**pw**$(Q)$: Max. number of marked vertices.

**pw**$(G)$: Min. pw$(Q)$ over all path-decompositions.

# Computing the Pathwidth

## Pathwidth problem

Pathwidth

| | |
|---|---|
| Input: | Graph $G$, $k \in \mathbb{N}$. |
| Question: | $\mathrm{pw}(\alpha) \leq k$? |

# Computing the Pathwidth

## Pathwidth problem

Pathwidth

| | |
|---|---|
| Input: | Graph $G$, $k \in \mathbb{N}$. |
| Question: | $\mathrm{pw}(\alpha) \leq k$? |

MinPathwidth denotes the corresponding minimisation problem.

# Computing the Pathwidth

## Pathwidth problem

Pathwidth

| | |
|---|---|
| Input: | Graph $G$, $k \in \mathbb{N}$. |
| Question: | $\text{pw}(\alpha) \leq k$? |

MinPathwidth denotes the corresponding minimisation problem.

## Known Results

- Pathwidth is NP-complete.
- Pathwidth $\in$ FPT (w.r.t. $k$).
- Exact exponential algorithms, linear fpt-algorithms, approximation algorithms...

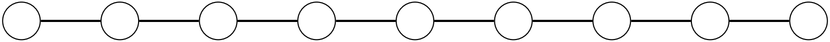
# Loc $\leq$ Pathwidth

$\alpha = \texttt{c a b a c a b a c}$

# Loc $\leq$ Pathwidth

$\alpha = \texttt{c a b a c a b a c}$

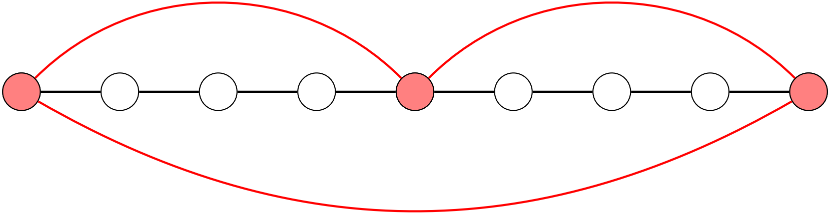
$\mathcal{G}_\alpha$:



# Loc $\leq$ Pathwidth

$\alpha = \texttt{c a b a c a b a c}$

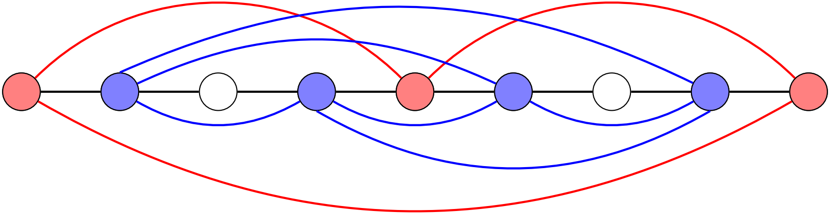
$\mathcal{G}_\alpha$:



# Loc ≤ Pathwidth

$\alpha = \texttt{c a b a c a b a c}$

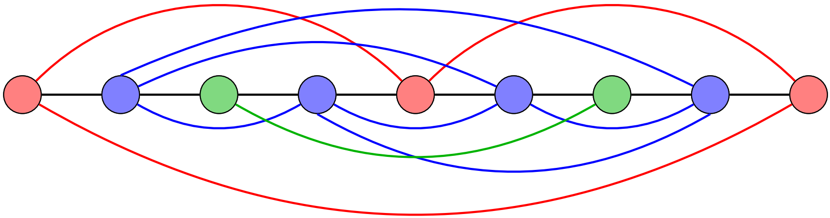
$\mathcal{G}_\alpha$:



# Loc $\leq$ Pathwidth

$\alpha = \mathtt{c\ a\ b\ a\ c\ a\ b\ a\ c}$

$\mathcal{G}_\alpha$:



# Results

**Lemma**

$\mathsf{loc}(\alpha) \leq \mathsf{pw}(\mathcal{G}_\alpha) \leq 2\,\mathsf{loc}(\alpha)$.

# Results

**Lemma**

$\mathsf{loc}(\alpha) \leq \mathsf{pw}(\mathcal{G}_\alpha) \leq 2\,\mathsf{loc}(\alpha)$.

**Lemma**

$\exists\alpha : \mathsf{pw}(\mathcal{G}_\alpha) = 2\,\mathsf{loc}(\alpha)$,

$\exists\beta : \mathsf{loc}(\beta) = \mathsf{pw}(\mathcal{G}_\beta)$.

# Results

### Lemma

$\mathsf{loc}(\alpha) \leq \mathsf{pw}(\mathcal{G}_\alpha) \leq 2\,\mathsf{loc}(\alpha)$.

### Lemma

$\exists \alpha : \mathsf{pw}(\mathcal{G}_\alpha) = 2\,\mathsf{loc}(\alpha)$,

$\exists \beta : \mathsf{loc}(\beta) = \mathsf{pw}(\mathcal{G}_\beta)$.

### Theorem

There is an $O(\sqrt{\log(\mathrm{opt})}\log(n))$-approx. algo. for MinLoc.

# Consequences for Cutwidth

### MinCutwidth

Leighton and Rao, JACM 1999: $O(\log(n)\log(n))$-approximation.

(Based on more general approximation techniques for edge-seperators)

# Consequences for Cutwidth

## MinCutwidth

Leighton and Rao, JACM 1999: $O(\log(n)\log(n))$-approximation.

(Based on more general approximation techniques for edge-seperators)

## MinPathwidth

Feige et al., SICOMP 2008: $O(\sqrt{\log(opt)}\log(n))$-approximation.

(Based on more general approximation techniques for vertex-seperators)

# Consequences for Cutwidth

### MinCutwidth

Leighton and Rao, JACM 1999: $O(\log(n)\log(n))$-approximation.

(Based on more general approximation techniques for edge-seperators)

### MinPathwidth

Feige et al., SICOMP 2008: $O(\sqrt{\log(\text{opt})}\log(n))$-approximation.

(Based on more general approximation techniques for vertex-seperators)

### MinCutwidth $\leq$ MinLoc $\leq$ MinPathwidth

There is an $O(\sqrt{\log(\text{opt})}\log(h))$-approximation algorithm for MinCutwidth on multigraphs with $h$ edges.

# Direct Reduction: MinCutwidth $\leq$ MinPathwidth