Subsequences with Generalised Gap Constraints

Upper and Lower Complexity Bounds

Florin Manea [1] *Jonas Richardsen* [1] Markus L. Schmid [2]

[1]Computer Science Department and CIDAS, Universität Göttingen, Germany
[2]Humboldt-Universität zu Berlin, Germany

June 27, 2024

Subsequences

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$

$$p = \quad x \quad y \quad z \quad y \quad x$$

$$w = \quad z \quad x \quad y \quad x \quad z \quad x \quad y \quad y \quad x$$

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$

$p =$ x y z y x

$w =$ z **x** **y** x **z** x **y** y **x**

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$

$$p = \quad \mathtt{x} \quad \mathtt{y} \quad \mathtt{z} \quad \mathtt{y} \quad \mathtt{x}$$

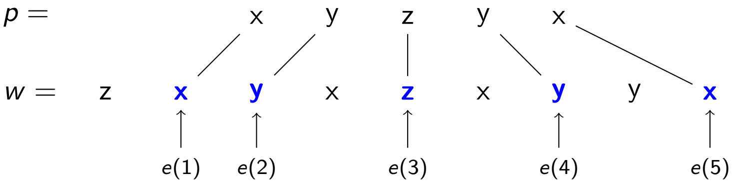
$$w = \quad \mathtt{z} \quad \mathbf{x} \quad \mathbf{y} \quad \mathtt{x} \quad \mathbf{z} \quad \mathtt{x} \quad \mathbf{y} \quad \mathtt{y} \quad \mathbf{x}$$

$w =$	z	**x**	**y**	x	**z**	x	**y**	y	**x**
		$\uparrow$	$\uparrow$		$\uparrow$		$\uparrow$		$\uparrow$
		$e(1)$	$e(2)$		$e(3)$		$e(4)$		$e(5)$

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$



Notation: $p \preceq_e w$ with $e : [m] \to [n]$ (or just $p \preceq w$)

Subsequences

Given:

- A text w with $|w| = n$
- A pattern p with $|p| = m$

$$\begin{array}{cccccccccc}
p = & & & \mathrm{x} & \mathrm{y} & \mathrm{z} & \mathrm{y} & \mathrm{x} & & \\
 & & & & & & & & & \\
w = & \mathrm{z} & \mathbf{x} & \mathbf{y} & \mathrm{x} & \mathbf{z} & \mathrm{x} & \mathbf{y} & \mathrm{y} & \mathbf{x} \\
 & & \uparrow & \uparrow & & \uparrow & & \uparrow & & \uparrow \\
 & & e(1) & e(2) & & e(3) & & e(4) & & e(5)
\end{array}$$

Notation: $p \preceq_e w$ with $e : [m] \to [n]$ (or just $p \preceq w$)

Properties of e:

- $e(1) < e(2) < \cdots < e(m)$
- $w[e(i)] = p[i]$

Gaps in subsequences

Gaps in subsequences

Definition

Given positions $i, j \in [m], i < j$, define

$$\mathrm{gap}_{w,e}[i,j] := w[e(i)+1..e(j)-1]$$

as the gap *between* the embedding of the i-th and j-th character.

Gaps in subsequences

Definition

Given positions $i, j \in [m], i < j$, define

$$\text{gap}_{w,e}[i,j] := w[e(i)+1..e(j)-1]$$

as the gap *between* the embedding of the i-th and j-th character.

Example:

$p =$ x y z y x

$w =$ z x y x z x y y x

Gaps in subsequences

Definition

Given positions $i, j \in [m], i < j$, define

$$\mathrm{gap}_{w,e}[i,j] := w[e(i)+1..e(j)-1]$$

as the gap *between* the embedding of the i-th and j-th character.

Example:

$p =$ x y z y x

$w =$ z x y **x** z x y y x

$$\mathrm{gap}_{w,e}[2,3] = \mathrm{x}$$

Gaps in subsequences

Definition

Given positions $i, j \in [m], i < j$, define

$$\mathrm{gap}_{w,e}[i,j] := w[e(i)+1..e(j)-1]$$

as the gap *between* the embedding of the i-th and j-th character.

Example:

$p =$ x y z y x

$w =$ z x y **x** **z** **x** y y x

$$\mathrm{gap}_{w,e}[2,4] = \mathrm{xzx}$$

Gaps in subsequences

Definition

Given positions $i, j \in [m], i < j$, define

$$\mathrm{gap}_{w,e}[i,j] := w[e(i) + 1..e(j) - 1]$$

as the gap *between* the embedding of the i-th and j-th character.

Example:

$p =$ x y z y x

$w =$ z x y x z x y y x

$$\mathrm{gap}_{w,e}[1,2] = \varepsilon$$

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\mathsf{gap}_{w,e}[i, j] \in L.$$

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\text{gap}_{w,e}[i, j] \in L.$$

Types of Gap-Constraints:

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\mathrm{gap}_{w,e}[i, j] \in L.$$

Types of Gap-Constraints:

- *regular constraint:* $L \in \mathrm{REG}$

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\mathrm{gap}_{w,e}[i, j] \in L.$$

Types of Gap-Constraints:

- *regular constraint:* $L \in \mathrm{REG}$, $\mathrm{size}(C) = \mathrm{size}(A) = \mathcal{O}(\mathrm{states}(A)|\Sigma|)$ with A being a DFA accepting L

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\text{gap}_{w,e}[i, j] \in L.$$

Types of Gap-Constraints:

- *regular constraint:* $L \in \text{REG}$, $\text{size}(C) = \text{size}(A) = \mathcal{O}(\text{states}(A)|\Sigma|)$ with A being a DFA accepting L
- *semilinear length constraint:* $L = \{w \in \Sigma^* \mid |w| \in S\}$ with S a semilinear set

Gap-Constraints

Definition

A gap constraint is defined as a triple $C = (i, j, L)$ with $i, j \in [m], i < j$ and $L \subseteq \Sigma^*$. An embedding e is said to satisfy the constraint if and only if

$$\mathrm{gap}_{w,e}[i, j] \in L.$$

Types of Gap-Constraints:

- *regular constraint:* $L \in \mathrm{REG}$, $\mathrm{size}(C) = \mathrm{size}(A) = \mathcal{O}(\mathrm{states}(A)|\Sigma|)$ with A being a DFA accepting L
- *semilinear length constraint:* $L = \{w \in \Sigma^* \mid |w| \in S\}$ with S a semilinear set, $\mathrm{size}(C) = \mathrm{size}(S)$

Semilinear sets

Definition

A subset $L \subseteq \mathbb{N}$ is called *linear*, if it is of the form

$$L = L(x_0; x_1, \ldots, x_m) = \{x_0 + \sum_{i=1}^{m} k_i x_i \mid k_1, \ldots, k_m \in \mathbb{N}_0\}$$

with $x_0 \in \mathbb{N}_0, x_1, \ldots, x_k \in \mathbb{N}$. A *semilinear* set is a finite union of linear sets.

Size of a (semi-)linear set:

- A linear set $L = L(x_0; x_1, \ldots, x_m)$ has size $\text{size}(L) = m + 1$.
- A semilinear set $L = \bigcup_{i=1}^{k} L_i$ with L_i linear sets has size $\text{size}(L) = \sum_{i=1}^{k} \text{size}(L_i)$.

The Matching Problem (MATCH)

Input:

- A text w with $|w| = n$
- A pattern p with $|p| = m$
- A set of constraints $\mathcal{C} = \{C_1, \ldots, C_k\}$

The Matching Problem (Match)

Input:

- A text w with $|w| = n$
- A pattern p with $|p| = m$
- A set of constraints $\mathcal{C} = \{C_1, \ldots, C_k\}$,
  $\mathsf{gapsize}(\mathcal{C}) := \max_{C \in \mathcal{C}}(\mathsf{size}(C))$

The Matching Problem (MATCH)

Input:

- A text w with $|w| = n$
- A pattern p with $|p| = m$
- A set of constraints $\mathcal{C} = \{C_1, \ldots, C_k\}$,
  $\mathrm{gapsize}(\mathcal{C}) := \max_{C \in \mathcal{C}}(\mathrm{size}(C))$

Question: Is p a $\mathcal{C}$-subsequence of w

The Matching Problem (MATCH)

Input:

- A text w with $|w| = n$
- A pattern p with $|p| = m$
- A set of constraints $\mathcal{C} = \{C_1, \ldots, C_k\}$,
  $\mathrm{gapsize}(\mathcal{C}) := \max_{C \in \mathcal{C}}(\mathrm{size}(C))$

Question: Is p a $\mathcal{C}$-subsequence of w, i.e., is there an embedding $e : [m] \to [n]$ with

- $p \preceq_e w$ and
- e satisfies C_ℓ for all $\ell \in [k]$?

The Matching Problem (Match)

Input:

- A text w with $|w| = n$
- A pattern p with $|p| = m$
- A set of constraints $\mathcal{C} = \{C_1, \ldots, C_k\}$,
  $\mathsf{gapsize}(\mathcal{C}) := \max_{C \in \mathcal{C}}(\mathsf{size}(C))$

Question: Is p a $\mathcal{C}$-subsequence of w, i.e., is there an embedding $e : [m] \to [n]$ with

- $p \preceq_e w$ and
- e satisfies C_ℓ for all $\ell \in [k]$?

We use $\textsc{Match}_{\mathsf{REG}}$ and $\textsc{Match}_{\mathsf{SLS}}$ to denote the variants of the matching problem with regular and semilinear length constraints respectively.

Polynomial solutions for $\textsc{Match}$ with constant $|\mathcal{C}|$

$\textsc{Match}_{\mathsf{REG}}$: Construct an NFA to solve the problem:

Theorem

$\textsc{Match}_{\mathsf{REG}}$ *can be solved in polynomial time for constant* $|\mathcal{C}|$ *and is fixed parameter tractable for the combined parameter* $(|p|, \mathsf{gapsize}(\mathcal{C}))$.

$\textsc{Match}_{\mathsf{SLS}}$: Enumerate all possible partial embeddings of the positions that have a constraint, then "fill the gaps":

Theorem

$\textsc{Match}_{\mathsf{SLS}}$ *can be solved in polynomial time for constant* $|\mathcal{C}|$.

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Definition (k-$\textsc{Clique}$)

Given a graph $G = (V, E)$ with n vertices, decide whether there is a subset $K \subseteq V$ of k vertices that are pairwise adjacent.

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Definition (k-$\textsc{Clique}$)

Given a graph $G = (V, E)$ with n vertices, decide whether there is a subset $K \subseteq V$ of k vertices that are pairwise adjacent.

Alternative question: Is the $k \times k$ matrix containing only 1's a principal submatrix of the adjacency matrix A of G?

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Definition (k-$\textsc{Clique}$)

Given a graph $G = (V, E)$ with n vertices, decide whether there is a subset $K \subseteq V$ of k vertices that are pairwise adjacent.

Alternative question: Is the $k \times k$ matrix containing only 1's a principal submatrix of the adjacency matrix A of G?

Example:

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Definition (k-$\textsc{Clique}$)

Given a graph $G = (V, E)$ with n vertices, decide whether there is a subset $K \subseteq V$ of k vertices that are pairwise adjacent.

Alternative question: Is the $k \times k$ matrix containing only 1's a principal submatrix of the adjacency matrix A of G?

Example:

$$A = \begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

Definition (k-Clique)

Given a graph $G = (V, E)$ with n vertices, decide whether there is a subset $K \subseteq V$ of k vertices that are pairwise adjacent.

Alternative question: Is the $k \times k$ matrix containing only 1's a principal submatrix of the adjacency matrix A of G?

Example:

$$A = \begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix} \implies K = \{v_1, v_2, v_4, v_6\} \text{ is a 4-Clique}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \ldots a_{nn}0$.

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \dots a_{nn}0$.
- *Anchor:* Enforce $e(1) = 1$ with the constraint $(1, k^2 + 2, L(n^2))$.

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \dots a_{nn}0$.
- *Anchor:* Enforce $e(1) = 1$ with the constraint $(1, k^2 + 2, L(n^2))$.
- Constrain the gap between the anchor and each element on the diagonal to be a multiple of $n + 1$.

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \ldots a_{nn}0$.
- *Anchor:* Enforce $e(1) = 1$ with the constraint $(1, k^2 + 2, L(n^2))$.
- Constrain the gap between the anchor and each element on the diagonal to be a multiple of $n + 1$.
- Constrain the gap between elements on subsequent rows and in the same column to be one smaller than a multiple of n.

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \ldots a_{nn}0$.
- *Anchor:* Enforce $e(1) = 1$ with the constraint $(1, k^2 + 2, L(n^2))$.
- Constrain the gap between the anchor and each element on the diagonal to be a multiple of $n + 1$.
- Constrain the gap between elements on subsequent rows and in the same column to be one smaller than a multiple of n.
- Constrain the gap between the first and last element of each row to be smaller than $n - 1$.

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}$$

Hardness of $\textsc{Match}_{\mathsf{SLS}}$

- Pattern $p = 01^{k \cdot k}0$, Text $w = 0a_{11} \dots a_{nn}0$.
- *Anchor:* Enforce $e(1) = 1$ with the constraint $(1, k^2 + 2, L(n^2))$.
- Constrain the gap between the anchor and each element on the diagonal to be a multiple of $n + 1$.
- Constrain the gap between elements on subsequent rows and in the same column to be one smaller than a multiple of n.
- Constrain the gap between the first and last element of each row to be smaller than $n - 1$.

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}$$

Theorem

$\textsc{Match}_{\mathsf{SLS}}$ *parameterised by* $|p|$ *is* $W[1]$*-hard, even for constant* gapsize$(\mathcal{C})$ *and binary alphabet* Σ*.*

Hardness of $\textsc{Match}_{\mathsf{REG}}$

Definition (1-in-3-3SAT)

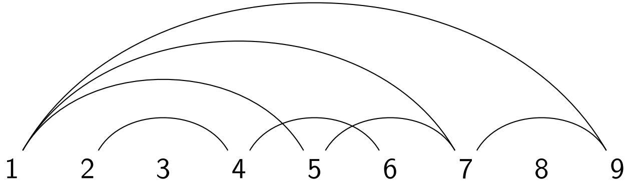
Given a set of variables $A = \{x_1, \ldots, x_n\}$ and clauses $c_1, \ldots, c_m \subseteq A$ with $|c_j| = 3$, find a subset $B \subseteq A$, such that $|c_i \cap B| = 1$ for every $j \in [m]$.

- Pattern $p = (\mathtt{b}\#)^n(\mathtt{b}\#)^m$
- Text $w = (\mathtt{bb}\#)^n(\mathtt{bbb}\#)^m$
- First part: decide $x_i \in B$ for $i \in [n]$.
- Second part: decide which of the 3 variables from c_j is in B for $j \in [m]$ (need some ordering on the variables).
- Use constraints to ensure that assignments are consistent.

Theorem

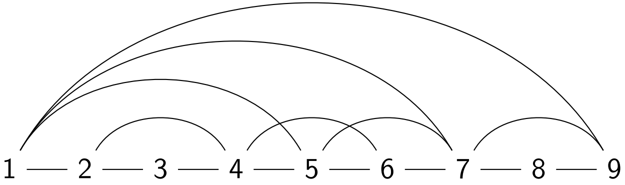
$\textsc{Match}_{\mathsf{REG}}$ is NP-complete, even for binary alphabet Σ and with gap-constraints that can be represented by DFAs with at most 8 states.

The Graph Structure of Constraints



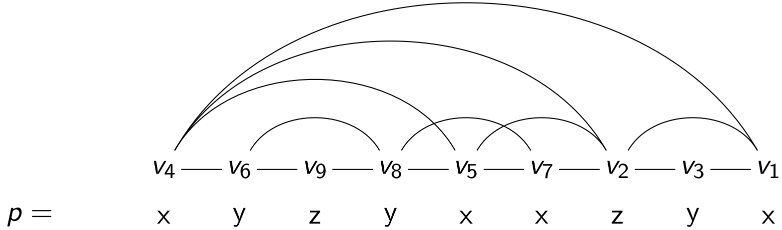
$p =$ x y z y x x z y x

The Graph Structure of Constraints



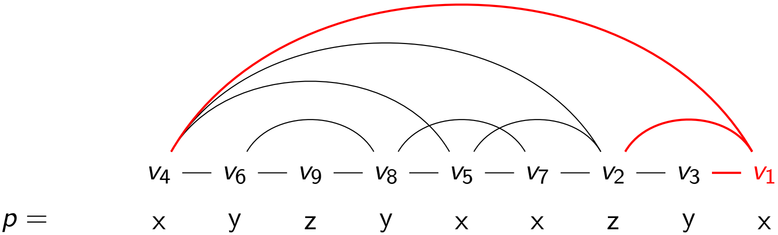
$p =$ x y z y x x z y x

The Graph Structure of Constraints



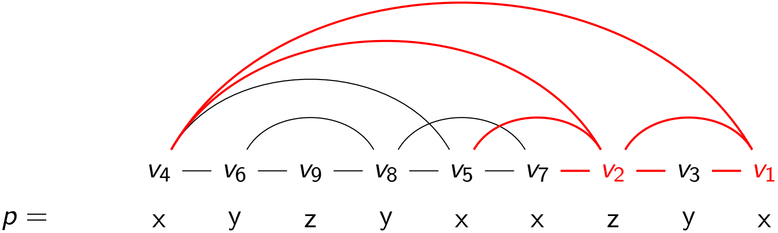
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



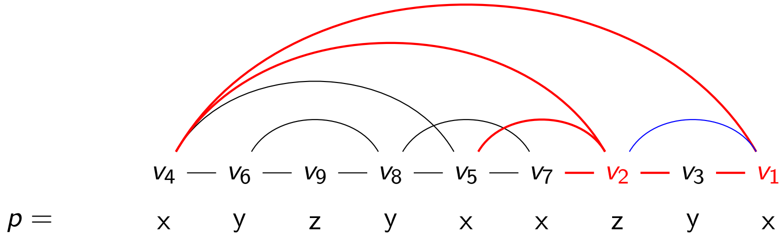
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



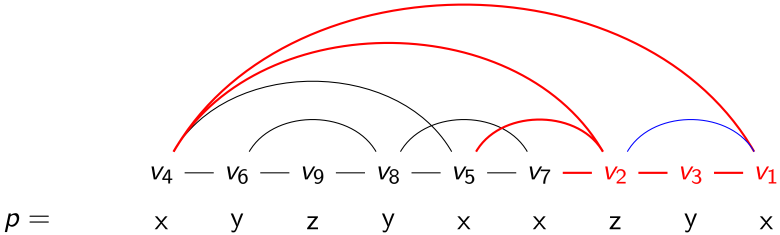
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



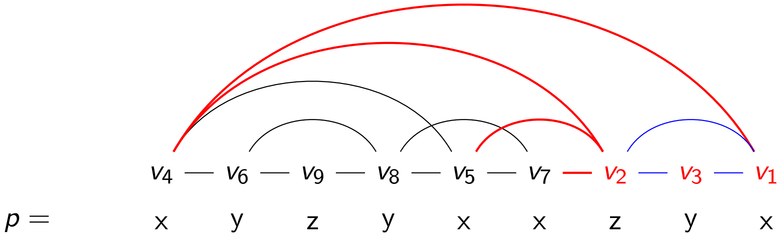
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



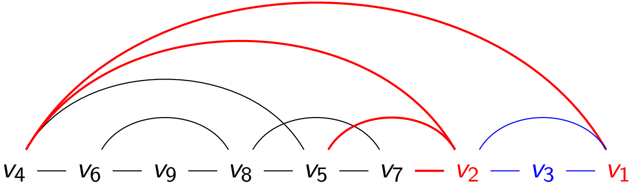
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \dots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

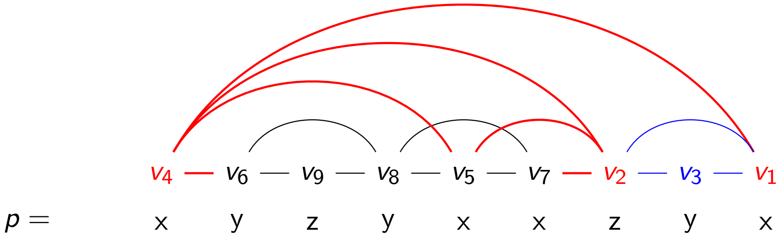
The Graph Structure of Constraints



$p =$ x y z y x x z y x

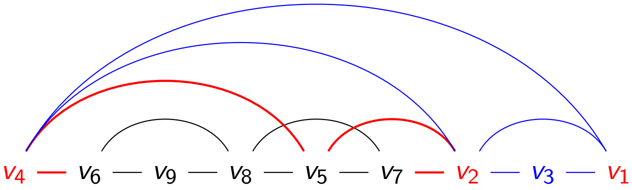
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

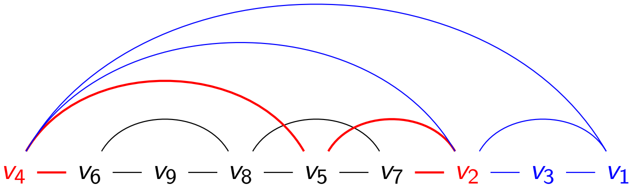
The Graph Structure of Constraints



$p =$ x y z y x x z y x

- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

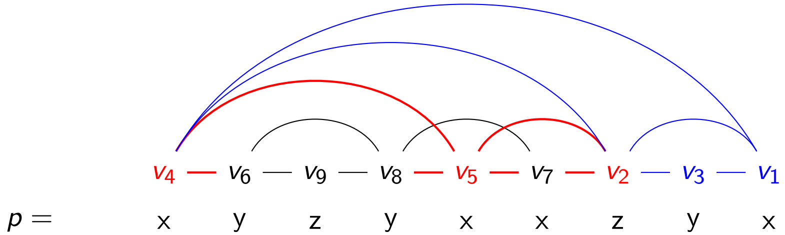
The Graph Structure of Constraints



$p =$ x y z y x x z y x

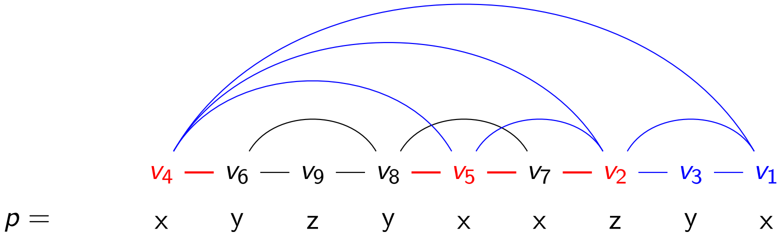
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



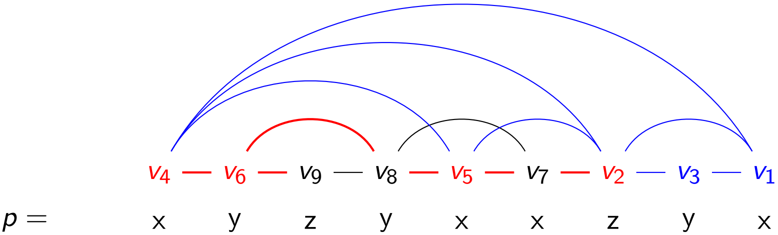
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



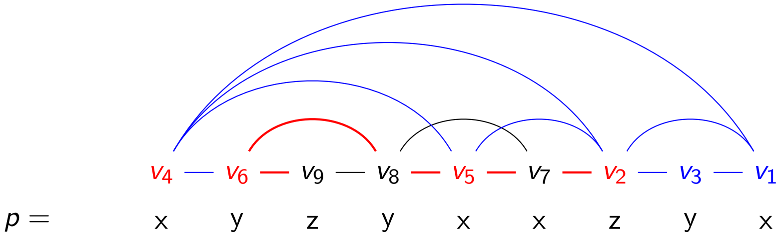
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



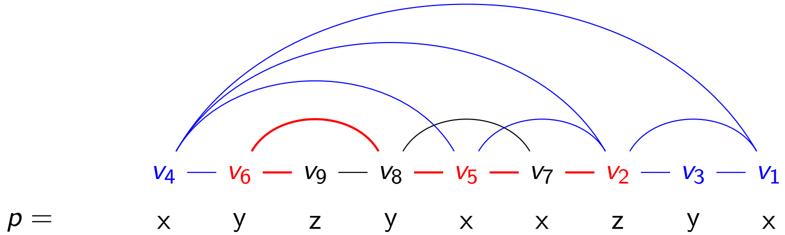
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



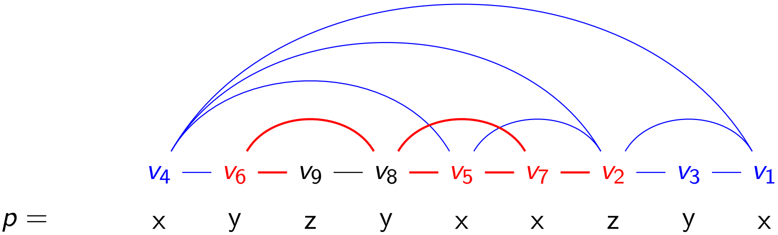
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



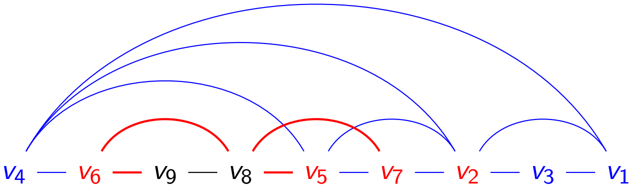
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

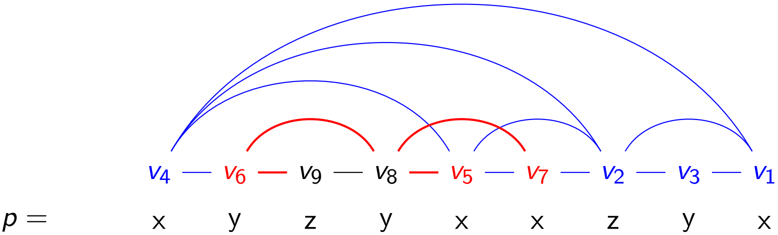
The Graph Structure of Constraints



$p =$ x y z y x x z y x

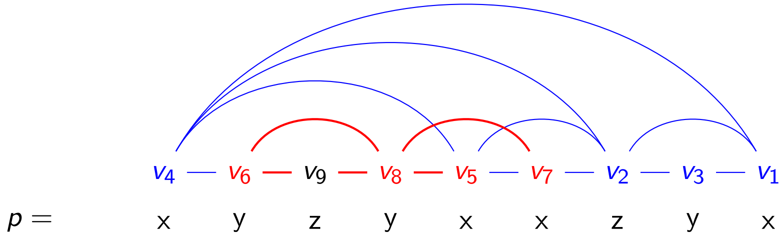
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \dots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



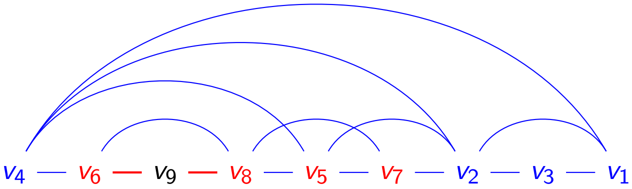
- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

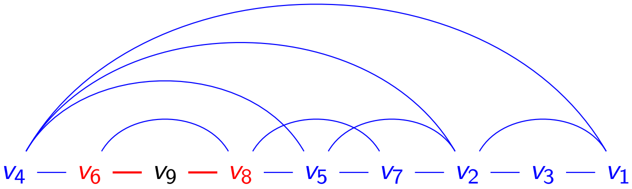
The Graph Structure of Constraints



$p =$ x y z y x x z y x

- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

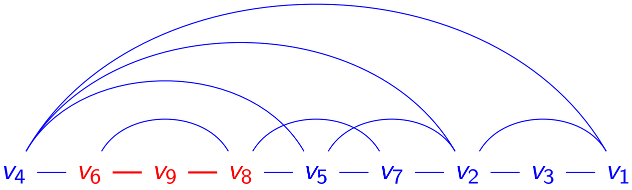
The Graph Structure of Constraints



$p =$ x y z y x x z y x

- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

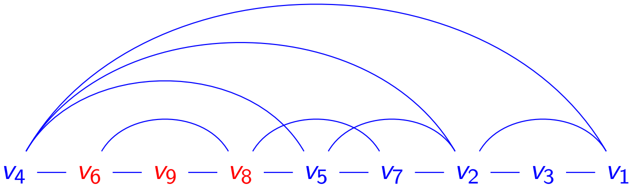
The Graph Structure of Constraints



$p =$ x y z y x x z y x

- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

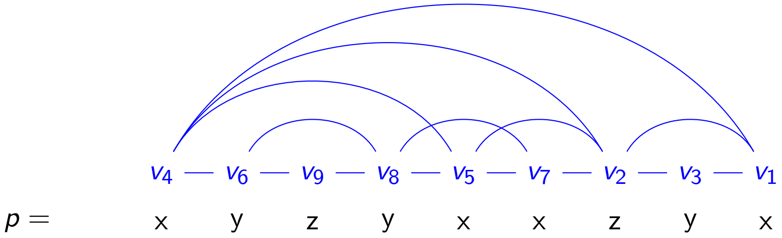
The Graph Structure of Constraints



$p =$ x y z y x x z y x

- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints



- Construct partial embeddings by adding the positions one-by-one in order $\sigma = (v_1, \ldots, v_m)$.
- At each time t, we compute all possible partial embeddings, but we only care about the the values $e(v_i)$, where $i \leq t$ and $(v_i, v_j) \in E$ for some $j > t$.
- When adding $e(v_t)$ to the embedding, combine all previously possible embeddings with all possible values of $e(v_t)$, then remove all embeddings that violate any constraint involving v_t.

The Graph Structure of Constraints

Definition (Vertex separation number)

Given a linear ordering $\sigma = (v_1, \ldots, v_n)$ of vertices in a graph, the vertex separation number of σ is the smallest number s such that, for each vertex v_i at most s vertices of $v_1, \ldots, v_{i-1}$ have some v_j with $j \geq i$ as neighbour. The vertex separation number of a graph is the minimum vertex separation number over all linear orderings of the graph.

The Graph Structure of Constraints

Definition (Vertex separation number)

Given a linear ordering $\sigma = (v_1, \ldots, v_n)$ of vertices in a graph, the vertex separation number of σ is the smallest number s such that, for each vertex v_i at most s vertices of $v_1, \ldots, v_{i-1}$ have some v_j with $j \geq i$ as neighbour. The vertex separation number of a graph is the minimum vertex separation number over all linear orderings of the graph.

Theorem

If the vertex separation number of the constraint graph is bound by k, $\textsc{Match}_{\mathsf{REG}}$ and $\textsc{Match}_{\mathsf{SLS}}$ can be solved in $\mathcal{O}(m^2 n^{k+1} + m^2 n^2 \log\log n)$ and $\mathcal{O}(m^2 n^{k+1})$ time respectively.

The Graph Structure of Constraints

Definition (Vertex separation number)

Given a linear ordering $\sigma = (v_1, \ldots, v_n)$ of vertices in a graph, the vertex separation number of σ is the smallest number s such that, for each vertex v_i at most s vertices of $v_1, \ldots, v_{i-1}$ have some v_j with $j \geq i$ as neighbour. The vertex separation number of a graph is the minimum vertex separation number over all linear orderings of the graph.

Theorem

If the vertex separation number of the constraint graph is bound by k, $\textsc{Match}_{\mathsf{REG}}$ and $\textsc{Match}_{\mathsf{SLS}}$ can be solved in $\mathcal{O}(m^2 n^{k+1} + m^2 n^2 \log\log n)$ and $\mathcal{O}(m^2 n^{k+1})$ time respectively.
Moreover, both variants are W[1]-hard (parameterized by the vertex separation number of the constraint graph).

The Graph Structure of Constraints

Definition (Vertex separation number)

Given a linear ordering $\sigma = (v_1, \ldots, v_n)$ of vertices in a graph, the vertex separation number of σ is the smallest number s such that, for each vertex v_i at most s vertices of $v_1, \ldots, v_{i-1}$ have some v_j with $j \geq i$ as neighbour. The vertex separation number of a graph is the minimum vertex separation number over all linear orderings of the graph.

Theorem

If the vertex separation number of the constraint graph is bound by k, $\textsc{Match}_{\mathsf{REG}}$ and $\textsc{Match}_{\mathsf{SLS}}$ can be solved in $\mathcal{O}(m^2 n^{k+1} + m^2 n^2 \log\log n)$ and $\mathcal{O}(m^2 n^{k+1})$ time respectively.
Moreover, both variants are W[1]-hard (parameterized by the vertex separation number of the constraint graph).

The second part is witnessed by the k-$\textsc{Clique}$ reduction from earlier.

The Interval Structure of Constraints

For $C = (i, j, L)$ define *interval*$(C) = [i, j - 1]$.

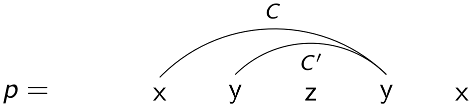
The Interval Structure of Constraints

For $C = (i, j, L)$ define $interval(C) = [i, j-1]$.
Let C and C' be two constraints.

The Interval Structure of Constraints

For $C = (i, j, L)$ define *interval*$(C) = [i, j-1]$.
Let C and C' be two constraints.

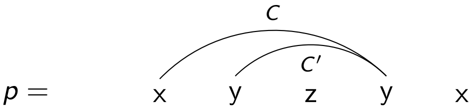
- *C contains C'* (or *C' is contained in C*) if *interval*$(C') \subsetneq$ *interval*(C)



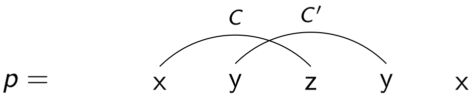
The Interval Structure of Constraints

For $C = (i, j, L)$ define *interval*$(C) = [i, j-1]$.
Let C and C' be two constraints.

- C *contains* C' (or C' *is contained in* C) if *interval*$(C') \subsetneq$ *interval*(C)



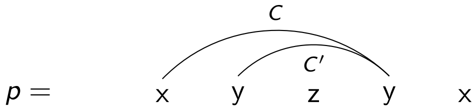
- C and C' *intersect* if *interval*$(C) \cap$ *interval*(C') is neither of *interval*(C), *interval*(C') or $\emptyset$.



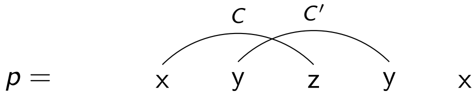
The Interval Structure of Constraints

For $C = (i, j, L)$ define *interval*$(C) = [i, j-1]$.
Let C and C' be two constraints.

- C *contains* C' (or C' *is contained in* C) if *interval*$(C') \subsetneq$ *interval*(C)



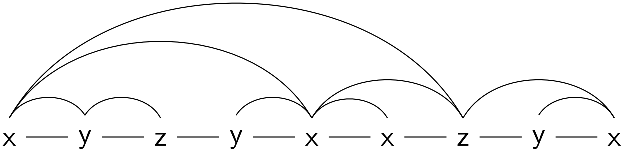
- C and C' *intersect* if *interval*$(C) \cap$ *interval*(C') is neither of *interval*(C), *interval*(C') or $\emptyset$.



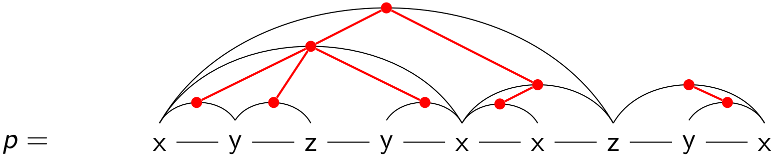
What about $\textsc{Match}$ with pairwise non-intersecting constraints?

Matching with non-intersecting constraints

$p =$

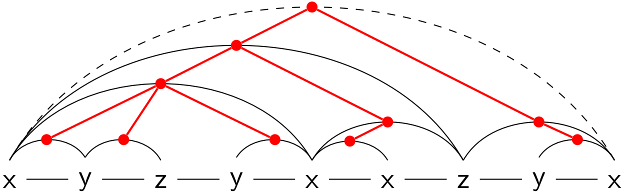


Matching with non-intersecting constraints



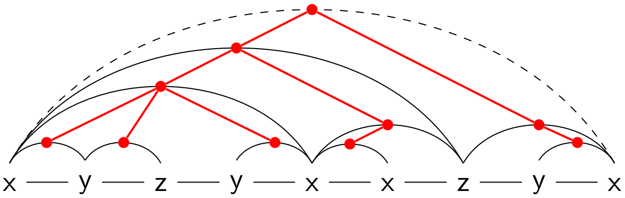
Matching with non-intersecting constraints

$p =$



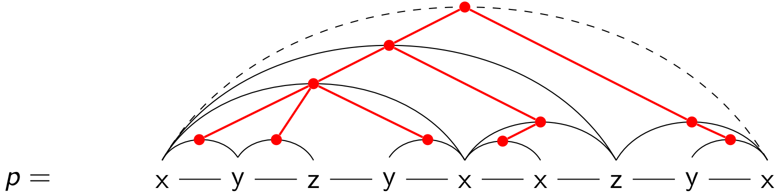
Matching with non-intersecting constraints

$p =$



We can recursively construct the partial embeddings of $p[i..j]$ for each constraint (i, j, L) bottom-up the tree.

Matching with non-intersecting constraints



We can recursively construct the partial embeddings of $p[i..j]$ for each constraint (i, j, L) bottom-up the tree.

Theorem

If constraints are pairwise non-intersecting, $\textsc{Match}_{\mathsf{REG}}$ *and* $\textsc{Match}_{\mathsf{SLS}}$ *can be solved in* $\mathcal{O}(n^\omega k + n^2 k \log\log n)$ *and* $\mathcal{O}(n^\omega k)$ *time respectively, where* $\mathcal{O}(n^\omega)$ *is the time needed to multiply two boolean matrices of size* $n \times n$.

Matching with non-intersecting constraints

Definition (3-OV)

Given three sets $A = \{\vec{a}_1, \ldots, \vec{a}_n\}, B = \{\vec{b}_1, \ldots, \vec{b}_n\}, C = \{\vec{c}_1, \ldots, \vec{c}_n\}$ of d-dimensional boolean vectors, are there indices $i, j, k \in [n]$, such that $\vec{a}_i \cdot \vec{b}_j \cdot \vec{c}_k = \vec{0}$?

Matching with non-intersecting constraints

Definition (3-OV)

Given three sets $A = \{\vec{a}_1, \ldots, \vec{a}_n\}, B = \{\vec{b}_1, \ldots, \vec{b}_n\}, C = \{\vec{c}_1, \ldots, \vec{c}_n\}$ of d-dimensional boolean vectors, are there indices $i, j, k \in [n]$, such that $\vec{a}_i \cdot \vec{b}_j \cdot \vec{c}_k = \vec{0}$?

$$p = \overline{\mathsf{C}}_p(\vec{a}_n) \ \ldots \ \overline{\mathsf{C}}_p(\vec{a}_i) \ \ldots \ \overline{\mathsf{C}}_p(\vec{a}_1) \quad \S \quad \mathsf{C}_p(\vec{a}_1) \ \ldots \ \mathsf{C}_p(\vec{a}_i) \ \ldots \ \mathsf{C}_p(\vec{a}_n)$$

$$w = \quad w_0 \ \overline{\mathsf{C}}_w(\vec{b}_n) \ldots \ \overline{\mathsf{C}}_w(\vec{b}_1) \ w_0 \quad \S \quad w_0 \ \mathsf{C}_w(\vec{c}_1) \ldots \ \mathsf{C}_w(\vec{c}_n) \ w_0$$

Matching with non-intersecting constraints

Definition (3-OV)

Given three sets $A = \{\vec{a}_1, \ldots, \vec{a}_n\}, B = \{\vec{b}_1, \ldots, \vec{b}_n\}, C = \{\vec{c}_1, \ldots, \vec{c}_n\}$ of d-dimensional boolean vectors, are there indices $i, j, k \in [n]$, such that $\vec{a}_i \cdot \vec{b}_j \cdot \vec{c}_k = \vec{0}$?

$$p = \overline{\mathsf{C}}_p(\vec{a}_n) \ \ldots \ \overline{\mathsf{C}}_p(\vec{a}_i) \ \ldots \ \overline{\mathsf{C}}_p(\vec{a}_1) \quad \S \quad \mathsf{C}_p(\vec{a}_1) \ \ldots \ \mathsf{C}_p(\vec{a}_i) \ \ldots \ \mathsf{C}_p(\vec{a}_n)$$

$$w = \quad w_0 \ \overline{\mathsf{C}}_w(\vec{b}_n) \ \ldots \ \overline{\mathsf{C}}_w(\vec{b}_1) \ w_0 \quad \S \quad w_0 \ \mathsf{C}_w(\vec{c}_1) \ \ldots \ \mathsf{C}_w(\vec{c}_n) \ w_0$$

Matching with non-intersecting constraints

Definition (3-OV)

Given three sets $A = \{\vec{a}_1, \ldots, \vec{a}_n\}, B = \{\vec{b}_1, \ldots, \vec{b}_n\}, C = \{\vec{c}_1, \ldots, \vec{c}_n\}$ of d-dimensional boolean vectors, are there indices $i, j, k \in [n]$, such that $\vec{a}_i \cdot \vec{b}_j \cdot \vec{c}_k = \vec{0}$?

$$p = \overline{\mathsf{C}}_p(\vec{a}_n) \;\ldots\; \overline{\mathsf{C}}_p(\vec{a}_i) \;\ldots\; \overline{\mathsf{C}}_p(\vec{a}_1) \quad \S \quad \mathsf{C}_p(\vec{a}_1) \;\ldots\; \mathsf{C}_p(\vec{a}_i) \;\ldots\; \mathsf{C}_p(\vec{a}_n)$$

$$w = \quad w_0 \;\; \overline{\mathsf{C}}_w(\vec{b}_n) \;\ldots\; \overline{\mathsf{C}}_w(\vec{b}_1) \;\; w_0 \quad \S \quad w_0 \;\; \mathsf{C}_w(\vec{c}_1) \;\ldots\; \mathsf{C}_w(\vec{c}_n) \;\; w_0$$

Theorem

Both variants of MATCH *with pairwise non-intersecting constraints cannot be solved in $\mathcal{O}(n^g k^h)$ time with $g + h < 3$, unless the Strong Exponential Time Hypothesis fails.*

Thank you for your attention!

Summary:

- *Problem:* Matching subsequences with gap constraints
- Two types of constraints: regular and semilinear length constraints
- Polynomial solutions for constant amount of constraints
- Hardness of the problem parameterized by the length of the pattern, witnessed by reductions from k-$\textsc{Clique}$ and 1-in-3-3-SAT.
- Graph structure of Constraints: Relation between complexity of the problem and vertex separation number of constraint graph
- Interval structure of Constraints: Efficient solution for the case of non-intersecting constraints, lower bound via fine-grained reduction from 3-OV.

Do you have any questions?