Pattern Matching with Variables: Fast Algorithms and New Hardness Results

Henning Fernau\textsuperscript{1} Florin Manea\textsuperscript{2} Robert Mercaş\textsuperscript{2,3} Markus L. Schmid\textsuperscript{1}

\textsuperscript{1}Trier University, Germany
\textsuperscript{2}Kiel University, Germany
\textsuperscript{3}King’s College, London, UK

STACS 2015
Patterns with Variables

Finite alphabet of terminals \( \Sigma = \{a, b, c, d\} \)

Set of variables \( X = \{x_1, x_2, x_3, \ldots\} \)

Patterns \( \alpha \in (\Sigma \cup X)^+ \)

Words \( w \in \Sigma^+ \)

Substitution \( h : X \rightarrow \Sigma^+ \)

\[ \alpha = y_1 \ldots y_n, \]
\[ h(\alpha) = h(y_1) \ldots h(y_n), \]
with \( h(a) = a, a \in \Sigma \).
Pattern Matching with Variables

pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w$. 
Pattern Matching with Variables

pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w$. 

$\alpha = x_1 x_2 x_1 x_3 x_2$

$w = \text{abbbabbbabababa}$
Pattern Matching with Variables

pattern $\alpha$ matches word $w \iff \exists$ substitution $h : h(\alpha) = w.$

$\alpha = abb x_2 abb x_3 x_2$

$w = abbbabaabbaaababa$
Pattern Matching with Variables

pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w$.

\[ \alpha = a b b b a a b b x_3 b a \]
\[ w = a b b b a a b b a a a b a b a b a b a b a \]
Pattern Matching with Variables

pattern $\alpha$ matches word $w$ $\iff \exists$ substitution $h : h(\alpha) = w$. 

$\alpha = a b b b a a b b a a a a b a b a b a b a b a$

$w = a b b b a a b b a a a a b a b a b a b a b a$
Pattern Matching with Variables

Pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w$. 

$$\alpha = x_1 a x_2 b x_2 x_1 x_2$$

$$w = bacbacbcbacbc$$
Pattern Matching with Variables

pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w.$

$\alpha = b a c b a x_2 b x_2 b a c b x_2$

$w = b a c b a c b c b a c b c$
Pattern Matching with Variables

Pattern $\alpha$ matches word $w$ $\iff$ $\exists$ substitution $h : h(\alpha) = w$. 

\[
\alpha = \text{bacbacbcbacbc} \\
w = \text{bacbacbcbacbc}
\]
Motivation

- Learning theory (inductive inference, PAC learning),
- language theory (pattern languages),
- combinatorics on words (word equations, unavoidable patterns, ambiguity of morphisms, equality sets),
- pattern matching (parameterised matching, (generalised) function matching),
- matchtest for regular expressions with backreferences (text editors (grep, emacs), programming language (Perl, Java, Python)),
- database theory.
Complexity

Matching Problem (MATCH)
Given a pattern \( \alpha \), a word \( w \). Does \( \alpha \) match \( w \) (i.e., \( \exists h : h(\alpha) = w \))?

- MATCH is (in general) NP-complete.
Complexity

Matching Problem (MATCH)
Given a pattern $\alpha$, a word $w$. Does $\alpha$ match $w$ (i.e., $\exists h : h(\alpha) = w$)?

- MATCH is (in general) NP-complete.

- **Bad news:** MATCH remains hard if numerical parameters are restricted (few exceptions):
  - MATCH $\in P$ if number of variables or word length bounded (trivial).
  - MATCH still hard if
    - alphabet size 2,
    - each variable has at most 2 occurrences,
    - $|h(x)| \leq 3$ for every $x$. 

Good news: Tractable if structure of patterns is restricted.
Complexity

Matching Problem (MATCH)
Given a pattern $\alpha$, a word $w$. Does $\alpha$ match $w$ (i.e., $\exists h : h(\alpha) = w$)?

- MATCH is (in general) NP-complete.

- **Bad news**: MATCH remains hard if numerical parameters are restricted (few exceptions):
  - MATCH $\in P$ if number of variables or word length bounded (trivial).
  - MATCH still hard if
    - alphabet size 2,
    - each variable has at most 2 occurrences,
    - $|h(x)| \leq 3$ for every $x$.

- **Good news**: Tractable if structure of patterns is restricted.
Notation

$\text{var}(\alpha)$ Set of variables occurring in pattern $\alpha$.

$|\alpha|_x$ Number of occurrences of variable $x$ in pattern $\alpha$. 
Structural Restrictions of Patterns

- **Regular Patterns:**
  \[ |\alpha|_x = 1, \ x \in \text{var}(\alpha). \]
  E. g., \( \alpha = abx_1x_2bx_3aaax_4b. \)
Structural Restrictions of Patterns

- **Regular Patterns:**
  \[ |\alpha|_x = 1, \ x \in \text{var}(\alpha). \]
  E. g., \( \alpha = abx_1x_2bx_3aaaax_4b. \)

- **Non-Cross Patterns:**
  \( \alpha = \ldots x \ldots y \ldots x \ldots \) is not possible.
  E. g., \( \alpha = x_1abaax_1ax_1x_2bx_2ax_2x_3x_3bbx_3ax_3 \)
Structural Restrictions of Patterns

- **k-Repeated-Variable Patterns:**
  \[ \left| \{ x \in \text{var}(\alpha) \mid |\alpha|_x \geq 2 \} \right| \leq k. \]
  
  E. g., \( \alpha = x_1abx_2ax_2ax_3bax_2bbx_4x_2x_5 \) is a 1-repeated-variable pattern.
Structural Restrictions of Patterns

- **k-Repeated-Variable Patterns:**
  \[ \left| \{ x \in \text{var}(\alpha) \mid |\alpha|_x \geq 2 \} \right| \leq k. \]
  E.g., \( \alpha = x_1abx_2ax_2ax_3bx_2bx_4x_2x_5 \) is a 1-repeated-variable pattern.

- **Pattern with Bounded Scope Coincidence Degree:**
  Scope (of \( x \)): shortest factor containing all occ. of \( x \),
  Scope coincidence degree: maximum number of coinciding scopes.

  \begin{align*}
  \alpha_1 &= \begin{array}{cccccccc}
  x_1 & x_2 & x_1 & x_3 & x_2 & x_3 & x_1 & x_2 & x_3 \\
  \end{array} \quad \text{scd}(\alpha_1) = 3 \\
  \alpha_2 &= \begin{array}{cccccccc}
  x_1 & x_2 & x_1 & x_1 & x_2 & x_3 & x_2 & x_3 & x_3 \\
  \end{array} \quad \text{scd}(\alpha_2) = 2
  \end{align*}
Known results: MATCH is in P for
- regular patterns \( O(|\alpha| + |w|) \),
- non-cross patterns \( O(|\alpha||w|^4) \),
- patterns with scd \( \leq k \) \( O(|\alpha||w|^{2(k+3)}(k + 2)^2) \).
Known results: MATCH is in P for

- regular patterns $\mathcal{O}(|\alpha| + |w|)$,
- non-cross patterns $\mathcal{O}(|\alpha||w|^4)$,
- patterns with scd $\leq k$ $\mathcal{O}(|\alpha||w|^{2(k+3)}(k + 2)^2)$.

Our contribution:

- Find (efficient) algorithms for these cases.
- Can we extend our algorithms to the injective case (i.e., different variables are replaced by different words)?
Lemma

MATCH for 1-repeated-variable patterns is solvable in $O(|w|^2)$.

Theorem

MATCH for $k$-repeated-variable patterns is solvable in $O\left(\frac{|w|^{2k}}{((k-1)!)^2}\right)$. 
Non-Cross Patterns

Dynamic programming approach!

\[ \alpha \text{ non-cross } \Rightarrow \]

\[ \alpha = w_0 \alpha_1 w_1 \alpha_2 \ldots \alpha_\ell w_\ell. \]

\[ \text{var}(\alpha_i) = \{x_i\}, \ w_i \in \Sigma^* \]
Non-Cross Patterns

Dynamic programming approach!

\( \alpha \) non-cross \( \Rightarrow \)

\( \alpha = w_0 \alpha_1 w_1 \alpha_2 \ldots \alpha_\ell w_\ell. \)

\( \text{var}(\alpha_i) = \{x_i\}, \ w_i \in \Sigma^* \)

Compute all sub-problems:

Does \( w_0 \alpha_1 w_1 \ldots w_{i-1} \alpha_i \) match \( w[1..j] \)？

\( 1 \leq i \leq \ell, 1 \leq j \leq |w| \)
Non-Cross Patterns

Case 1: $\alpha_i = x_i$

\[ w_0\alpha_1 w_1 \ldots w_{i-1} \alpha_i \]
\[ \downarrow \]
\[ w[1..j] \]
Non-Cross Patterns

Case 1: $\alpha_i = x_i$

\[ w_0\alpha_1 w_1 \ldots w_{i-1} x_i \]
\[ \downarrow \]
\[ w[1..j] \]
Non-Cross Patterns

Case 1: $\alpha_i = x_i$

\[
\begin{align*}
  w_0\alpha_1w_1 \ldots w_{i-1} & \quad x_i \\
  \downarrow & \\
  w[1..j] & \\
\end{align*}
\]

\[\iff\]

\[
\begin{align*}
  w_0\alpha_1w_1 \ldots w_{i-1} & \\
  \downarrow & \\
  w[1..j'] & \\
\end{align*}
\]
Non-Cross Patterns

Case 1: $\alpha_i = x_i$

\[
\begin{array}{c}
\alpha_1 \alpha_2 \ldots \alpha_{i-1} x_i \\
\downarrow \\
w[1..j]
\end{array}
\iff
\begin{array}{c}
\alpha_1 \alpha_2 \ldots \alpha_{i-1} x_i \\
\downarrow \\
w[j'+1..j]
\end{array}
\]
Non-Cross Patterns

Case 2a: $\alpha_i = (x_i)^k$  

$(x_i$ is mapped to primitive word $t$)

$w_0\alpha_1w_1 \ldots w_{i-1} \alpha_i$

$\downarrow$

$w[1..j]$
Non-Cross Patterns

Case 2a: $\alpha_i = (x_i)^k$  
(x_i is mapped to primitive word t)

\[
\begin{align*}
   &w_0\alpha_1w_1 \cdots w_{i-1} \ x_i x_i \cdots x_i \\
 \downarrow \\
   &w[1..j]
\end{align*}
\]
Non-Cross Patterns

Case 2a: \( \alpha_i = (x_i)^k \)  
\( (x_i \text{ is mapped to primitive word } t) \)

\[
\begin{align*}
    w_0 \alpha_1 w_1 \ldots w_{i-1} & \ x_i x_i \ldots x_i \\
    \downarrow & \\
    w[1..j] \\
\end{align*}
\]

\[\iff\]

\( \exists \text{ primitive word } t \text{ with } t^k \text{ suffix of } w[1..j] \text{ and } \)

\[
\begin{align*}
    w_0 \alpha_1 w_1 \ldots w_{i-1} & \\
    \downarrow & \\
    w[1..j - (k|t|)]
\end{align*}
\]
Non-Cross Patterns

Case 2a: \( \alpha_i = (x_i)^k \)

\( (x_i \text{ is mapped to primitive word } t) \)

\[
\begin{align*}
  w_0 \alpha_1 w_1 \ldots w_{i-1} & \quad x_i x_i \ldots x_i \\
  \downarrow & \\
  w[1..j] & 
\end{align*}
\]

\[
\iff 
\]

\( \exists \text{ primitive word } t \text{ with } t^k \text{ suffix of } w[1..j] \text{ and } \)

\[
\begin{align*}
  w_0 \alpha_1 w_1 \ldots w_{i-1} & \quad x_i x_i \ldots x_i \\
  \downarrow & \\
  w[1..j - (k|t|)] & \quad tt \ldots t
\end{align*}
\]
Non-Cross Patterns

Case 2a: Find all primitive $t$ such that $w[1..j]$ has $t^2$ as a suffix!

**Lemma (Crochemore, 1981)**

Primitive $u_1, u_2, u_3$, $|u_1| < |u_2| < |u_3|$, $w = w_i u_i u_i$, $1 \leq i \leq 3 \Rightarrow 2|u_1| < |u_3|$.

$\Rightarrow w$ has at most $2 \log |w|$ primitively rooted squares as suffix.
Non-Cross Patterns

Case 2a: Find all primitive $t$ such that $w[1..j]$ has $t^2$ as a suffix!

Lemma (Crochemore, 1981)

Definition: Primitive $u_1, u_2, u_3$, $|u_1| < |u_2| < |u_3|$, $w = w_i u_i u_i$, $1 \leq i \leq 3$ \Rightarrow $2|u_1| < |u_3|.$

\(\Rightarrow w\) has at most $2 \log |w|$ primitively rooted squares as suffix.

Lemma

We can compute in $O(n \log n)$ time all the sets $P_i = \{u \mid u \text{ primitive, } u^2 \text{ suffix of } w[1..i]\}, 1 \leq i \leq |w|.$

\(\Rightarrow\) Case 2a can be done efficiently.
Non-Cross Patterns

Case 2b: $\alpha_i = (x_i)^k$  

$(x_i$ is mapped to some word $t = v^{h+1}$)

\[ w_0\alpha_1w_1 \ldots w_{i-1} \ x_i x_i \ldots x_i \]

\[ \downarrow \]

\[ w[1..j] \]
Non-Cross Patterns

Case 2b: $\alpha_i = (x_i)^k$  

$(x_i$ is mapped to some word $t = v^{h+1}$)

$$w_0\alpha_1 w_1 \ldots w_{i-1} x_i x_i \ldots x_i$$

$$\downarrow$$

$$w[1..j]$$

$$\iff$$

$\exists$ primitive word $v$ with $v^k$ suffix of $w[1..j]$ and

$$w_0\alpha_1 w_1 \ldots w_{i-1} x_i x_i \ldots x_i$$

$$\downarrow$$

$$w[1..j - k|v|]$$

with $h(x_i) = v^h$
Non-Cross Patterns

Case 3: \( \alpha_i = x_i^{l_0} u_1 x_i^{l_1} u_2 \ldots x_i^{l_{p-1}} u_p x_i^{l_p} \) \( u_k \in \Sigma^+ \)

\[
\begin{align*}
w_0 \alpha_1 w_1 \ldots w_{i-1} & \quad \alpha_i \\
\downarrow \\
w[1..j]
\end{align*}
\]
Case 3: $\alpha_i = x_i^{\ell_0} u_1 x_i^{\ell_1} u_2 \ldots x_i^{\ell_p-1} u_p x_i^{\ell_p}$

$u_k \in \Sigma^+$

$w_0 \alpha_1 w_1 \ldots w_{i-1} x_i^{\ell_0} u_1 x_i^{\ell_1} u_2 \ldots x_i^{\ell_p-1} u_p x_i^{\ell_p}$

$\downarrow$

$w[1..j]$
Non-Cross Patterns

Case 3: \( \alpha_i = x_i^{\ell_0} u_1 x_i^{\ell_1} u_2 \ldots x_i^{\ell_{p-1}} u_p x_i^{\ell_p} \)

\[ u_k \in \Sigma^+ \]

\[
\begin{align*}
  w_0\alpha_1w_1 \ldots w_{i-1} & \quad x_i^{\ell_0} u_1 x_i^{\ell_1} u_2 \ldots x_i^{\ell_{p-1}} u_p x_i^{\ell_p} \\
  \downarrow & \\
  w[1..j]
\end{align*}
\]

- \( \ell_p \geq 2 \): proceed similar to Case 2 (more involved, details omitted).
- \( \ell_p = 1 \): find all primitive \( u_p t \) such that \( tu_p t \) is a suffix of \( w[1..j] \).
Generalisation of Crochemore’s result:

Lemma
For a fixed $v, w$ has $O(\log |w|)$ factors $uvu$ with $uv$ primitive as suffixes.

Lemma
For fixed $v, w$, we can compute in $O(n \log n)$ time all the sets
$R^*_i = \{u \mid uv \text{ primitive, } uvu \text{ suffix of } w[1..i]\}, 1 \leq i \leq |w|.$

⇒ Case 3 can be done efficiently.
Non-Cross Patterns

Theorem

**Match for non-cross patterns** is solvable in $O(|w|m \log |w|)$, where $m$ is the number of one-variable blocks of the pattern.

Theorem

**Match for patterns with scope coincidence degree of at most $k$** is solvable in $O\left(\frac{|w|^{2k}m}{((k-1)!)^2}\right)$, where $m$ is the number of one-variable blocks of the pattern.
**Injective MATCH**

**INJMATCH**: Like MATCH, but we are looking for an injective substitution $h$, i.e., $x \neq y \Rightarrow h(x) \neq h(y)$.

Can we use our (or other) MATCH-algorithms also for INJMATCH?

**INJMATCH** remains NP-complete for patterns for which MATCH is (trivially) in P.
Injective MATCH

**Theorem**
\[
\text{INJMatch} \text{ is } \text{NP-complete even for patterns } x_1x_2\ldots x_n, \ n \geq 1.
\]

We prove NP-completeness of the equivalent problem

**UNFACT**

*Instance:* A word \( w \) and an integer \( k \geq 1 \).

*Question:* \( w = u_1u_2\ldots u_{k'} \) with \( k' \geq k \) and \( u_i \neq u_j, \ 1 \leq i < j \leq k \)?

**Corollary**

\[
\text{INJMatch} \text{ is } \text{NP-complete for regular, non-cross, } k\text{-repeated-variable, bounded scd patterns.}
\]
Hardness of \textsc{InjMatch} - Proof Idea

\textbf{3D-Match}

\textit{Instance:} An integer $\ell \in \mathbb{N}$ and a set $S \subseteq \{(p, q, r) \mid 1 \leq p < \ell + 1 \leq q < 2\ell + 1 \leq r \leq 3\ell\}$.
\textit{Question:} Does there exist a subset $S'$ of $S$ with cardinality $\ell$ such that, for each two elements $(p, q, r), (p', q', r') \in S'$, $p \neq p'$, $q \neq q'$ and $r \neq r'$?
Hardness of \textsc{InjMatch} - Proof Idea

\textbf{3D-Match instance} $(S, \ell)$: $S = \{s_1, s_2, \ldots, s_k\}$
Transform every $s_i = (p_i, q_i, r_i)$, $1 \leq i \leq k$, into

\[ v_i = \star_i \; p_i \; a \; b_{i,1} \; b_{i,2} \; q_i \; a \; b_{i,3} \; b_{i,4} \; r_i \; a \; \diamond_i \]

$\star_i$, $\diamond_i$, $b_{i,j}$ have only one occurrence!
Hardness of **INJMATCH** - Proof Idea

**3D-MATCH** instance \((S, \ell)\): \(S = \{s_1, s_2, \ldots, s_k\}\)

Transform every \(s_i = (p_i, q_i, r_i), 1 \leq i \leq k\), into

\[
v_i = \star_i p_i \; a \; b_{i,1} \; b_{i,2} \; q_i \; a \; b_{i,3} \; b_{i,4} \; r_i \; a \; \diamond_i
\]

\(*_i, \diamond_i, b_{i,j}\) have only one occurrence!

Let \(S' \subseteq S\).

\[
(p_i, q_i, r_i) \notin S' \iff \star_i p_i \; a b_{i,1} \; b_{i,2} q_i \; a b_{i,3} \; b_{i,4} r_i \; a \Diamond_i
\]

\[
(p_i, q_i, r_i) \in S' \iff \star_i p_i a \; b_{i,1} b_{i,2} \; q_i a \; b_{i,3} b_{i,4} \; r_i a \; \Diamond_i
\]
Hardness of \textbf{INJMATCH} - Proof Idea

\textbf{3D-MATCH} instance \((S, \ell)\): \(S = \{s_1, s_2, \ldots, s_k\}\)
Transform every \(s_i = (p_i, q_i, r_i), 1 \leq i \leq k\), into

\[ v_i = \star_i p_i \ a \ b_{i,1} \ b_{i,2} \ q_i \ a \ b_{i,3} \ b_{i,4} \ r_i \ a \ \Diamond_i \]

\(\star_i, \Diamond_i, b_{i,j}\) have only one occurrence!

Let \(S' \subseteq S\).

\[ (p_i, q_i, r_i) \notin S' \iff \star_i p_i \ ab_{i,1} \ b_{i,2} q_i \ ab_{i,3} \ b_{i,4} r_i \ a \Diamond_i \]

\[ (p_i, q_i, r_i) \in S' \iff \star_i p_i a \ b_{i,1} b_{i,2} \ q_i a \ b_{i,3} b_{i,4} r_i a \Diamond_i \]

\(v = u_1 u_2 \ldots u_n\) with \(n = 7\ell + 6(k - \ell)\) and \(u_i \neq u_j, 1 \leq i < j \leq n\)

\(\iff\)

\(S'\) is a solution of \((S, \ell)\).
Alphabet Size

Our Reduction needs an unbounded alphabet!

Hardness of \textsc{InjMatch} for fixed alphabets is open, but...

\textbf{Theorem}

\textsc{InjMatch} (with constant alphabet) is NP-complete for regular, non-cross, \textit{k}-repeated-variable, bounded scd patterns.
Thank you very much for your attention.