PERMUTATION-CONSTRAINED COMMON STRING PARTITIONS WITH APPLICATIONS

Manuel Lafond

UNIVERSITÉ DE SHERBROOKE

Binhai Zhu

MONTANA STATE UNIVERSITY

In this talk

- Many genomic distances fit into a generic framework
- Permutation-Constrained String Partition (PCSP) solves genomic distances
- PCSP is FPT in (k + d)
  - k = # of blocks and d = max symbol occurrences
  - => many genomic distances are FPT in (k + d)

Genomic distances

- Given : two strings S and T, a set of allowed operations P
- Find : a minimum sequence of P operations to turn S into T

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T
  - example : P = {transposition} (swap two consecutive substrings)

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T
  - example : P = {transposition} (swap two consecutive substrings)

S = a b c d a d b c e

T = a a d c e b c d b

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T

  - example : P = {transposition} (swap two consecutive substrings)

$$S = a\ \underline{b\ c\ d}\ \underline{a\ d}\ b\ c\ e$$

$$\Downarrow$$

$$a\ a\ d\ b\ c\ d\ b\ c\ e$$

$$T = a\ a\ d\ c\ e\ b\ c\ d\ b$$

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T
  - example : P = {transposition} (swap two consecutive substrings)

$$\begin{array}{c}
S = a\ \underline{b\ c\ d}\ \underline{a\ d}\ b\ c\ e \\
\Downarrow \\
a\ a\ d\ \underline{b\ c\ d\ b}\ \underline{c\ e} \\
\Downarrow \\
T = a\ a\ d\ c\ e\ b\ c\ d\ b
\end{array}$$

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T
    - example : P = {inversion} (revert + change sign of a substring)

$$S = a^+ \, b^+ \, c^+ \, d^+ \, a^- \, d^+ \, b^+ \, c^-$$

$$T = a^+ \, b^+ \, a^+ \, d^- \, c^+ \, b^- \, d^- \, c^+$$

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T

  - example : P = {inversion} (revert + change sign of a substring)

$$S = a^+ \ b^+ \ \underline{c^+ \ d^+ \ a^-} \ d^+ \ b^+ \ c^-$$

$$\Downarrow$$

$$a^+ \ b^+ \ a^+ \ d^- \ c^- \ d^+ \ b^+ \ c^-$$

$$T = a^+ \ b^+ \ a^+ \ d^- \ c^+ \ b^- \ d^- \ c^+$$

Genomic distances

- **Given** : two strings S and T, a set of allowed operations P
- **Find** : a minimum sequence of P operations to turn S into T
  - example : P = {inversion} (revert + change sign of a substring)

$$S = a^+\ b^+\ \underline{c^+\ d^+\ a^-}\ d^+\ b^+\ c^-$$

$$\Downarrow$$

$$a^+\ b^+\ a^+\ d^-\ \underline{c^-\ d^+\ b^+\ c^-}$$

$$\Downarrow$$

$$T = a^+\ b^+\ a^+\ d^-\ c^+\ b^-\ d^-\ c^+$$

Many other operations

- Transpositions
- Block interchange (swap any two substrings)
- Inversions
- Unsigned inversions (reverse substring, don’t change sign)
- Flip (flip sign of substring, don’t reverse)
- k-cut (split into k substrings, permute them)
- k-inversion (choose k substrings, invert them all)

- + P could be any subset of these

Genomic distances

- Given : two strings S and T, a set of allowed operations P
- Find : a minimum sequence of P operations to turn S into T

- If S and T are permutations, most P are FPT in k = distance.
    - Transpositions is hard [Bulteau, Fertin, & Rusu, SIDMA12]
- All are FPT in k + |S|
- S and T are not permutations (and have unbounded size)!
- In general, FPT complexity unknown for most P.
- In this work : all FPT in $k + d$

Untouched blocks

- A minimum transformation scenario leaves some blocks of S and T “untouched”

$$S = a\ \underline{b\ c\ d}\ \underline{a\ d}\ b\ c\ e$$

$$\Downarrow$$

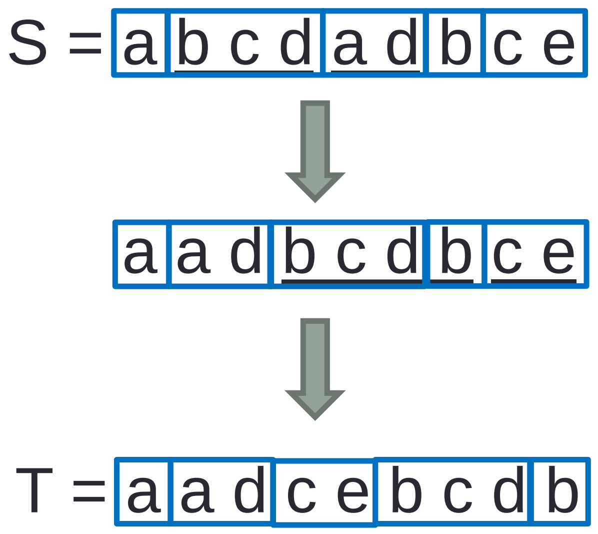
$$a\ a\ d\ \underline{b\ c\ d\ b}\ \underline{c\ e}$$

$$\Downarrow$$

$$T = a\ a\ d\ c\ e\ b\ c\ d\ b$$

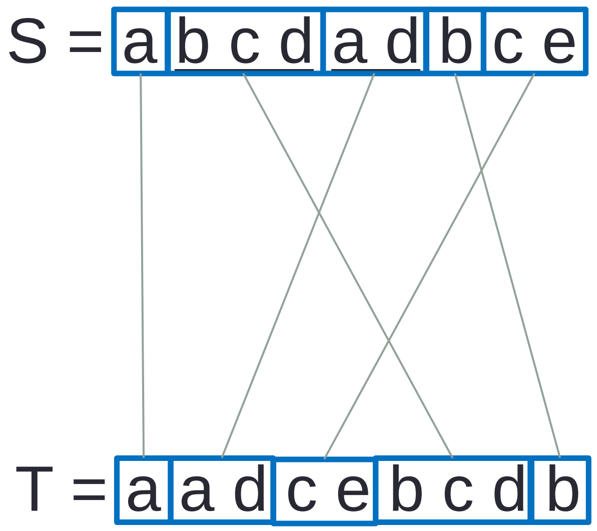
Untouched blocks

- A minimum transformation scenario leaves some blocks of S and T "untouched"


S = a | b c d | a d | b | c e
a | a d | b c d | b | c e
T = a | a d | c e | b c d | b

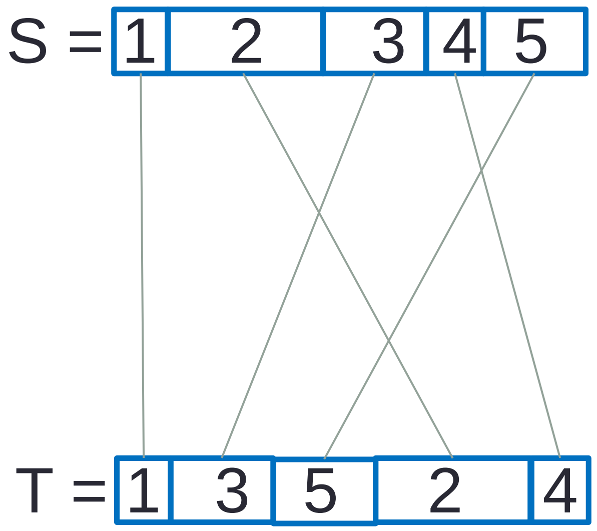
Untouched blocks

- A minimum transformation scenario leaves some blocks of S and T "untouched"



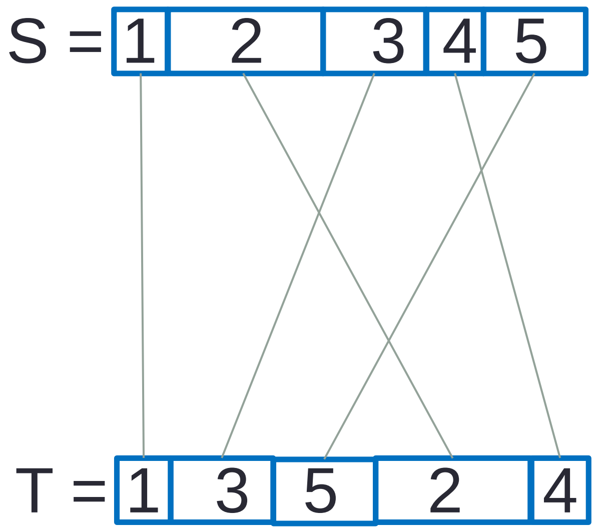
Untouched blocks

- A minimum transformation scenario leaves some blocks of S and T "untouched"



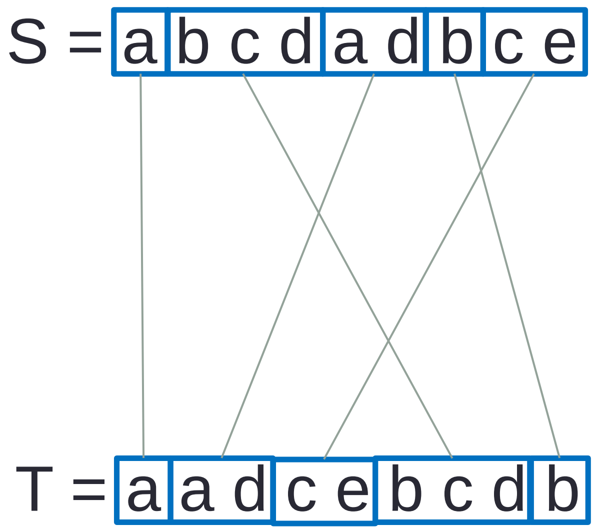
Untouched blocks

- A minimum transformation scenario leaves some blocks of S and T “untouched”
- If the matching blocks were known, we could reduce to the permutation variant.



Minimum Common String Partition (MCSP)

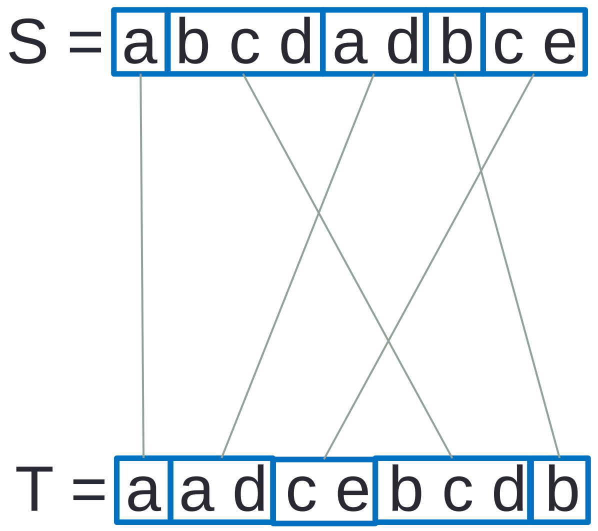
- **Given** : two strings S and T, integer k
- **Find** : a partition into at most k blocks of S and T that allow a perfect matching of equal blocks.



Minimum Common String Partition (MCSP)

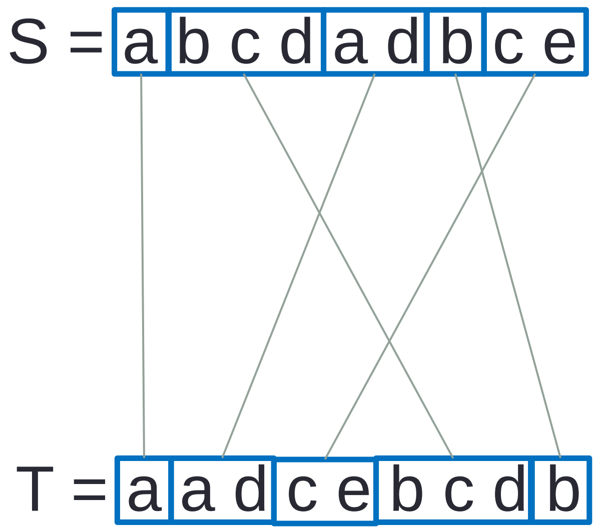
- Given : two strings S and T, integer k
- Find : a partition into at most k blocks of S and T that allow a perfect matching of equal blocks.

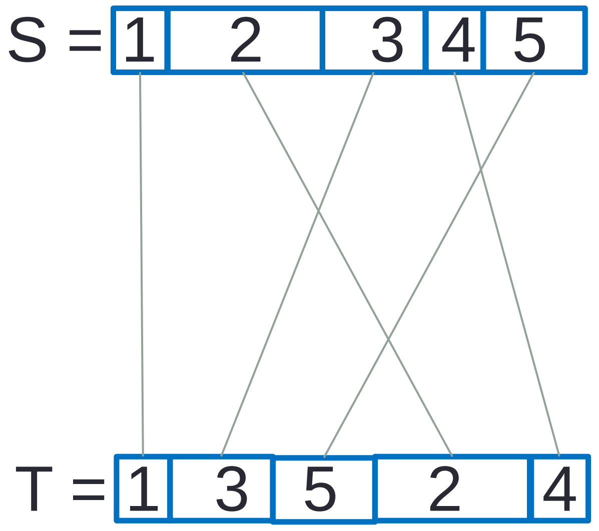
- MCSP is FPT in k [Bulteau & Komusiewicz, SODA2014]



An idea

- Find a MCSP in time $O(2^{k^2}n)$
- Assign each matched block a unique id
- Solve the resulting permutation instance





An idea

- Find a MCSP in time $O(2^{k^2}n)$
- Assign each matched block a unique id
- Solve the resulting permutation instance

- PROBLEM : a MCSP **might not correspond** to the blocks resulting from an optimal transformation sequence.

An idea

- Find a MCSP in time $O(2^{k^2} n)$
- Assign each matched block a unique id
- Solve the resulting permutation instance

- PROBLEM : a MCSP **might not correspond** to the blocks resulting from an optimal transformation sequence.
  - All MCSP need to be considered.
  - Worse: suboptimal MCSPs need to be considered, up to size p * k.
  - Bulteau & Komusiewicz's algorithm cannot list all those in FPT time.

Permutation-Constrained Common String Partition (PCSP)

- **Given** : two strings S and T, integer k, permutation **π** of [k]
- **Find** : a partition into k blocks of S and T, with a perfect matching that agrees with **π**.

$k = 5$

$\boldsymbol{\pi} : (1, 4, 2, 5, 3)$

S = a b c d a d b c e

T = a a d c e b c d b

Permutation-Constrained Common String Partition (PCSP)

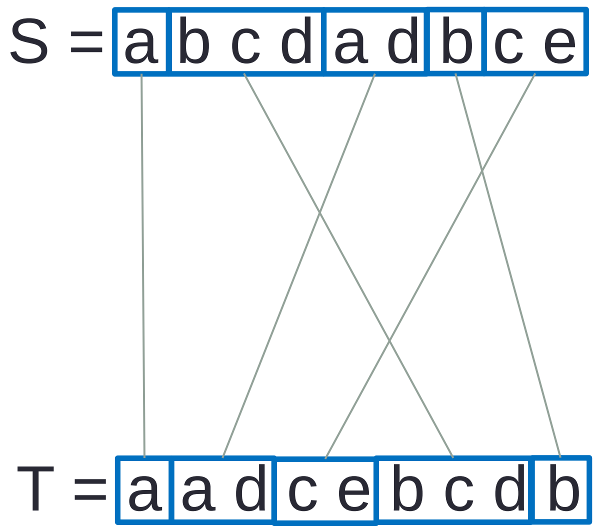
- **Given** : two strings S and T, integer k, permutation **π** of [k]
- **Find** : a partition into k blocks of S and T, with a perfect matching that agrees with **π**.

$k = 5$

$\boldsymbol{\pi} : (1, 4, 2, 5, 3)$

S = a b c d a d b c e

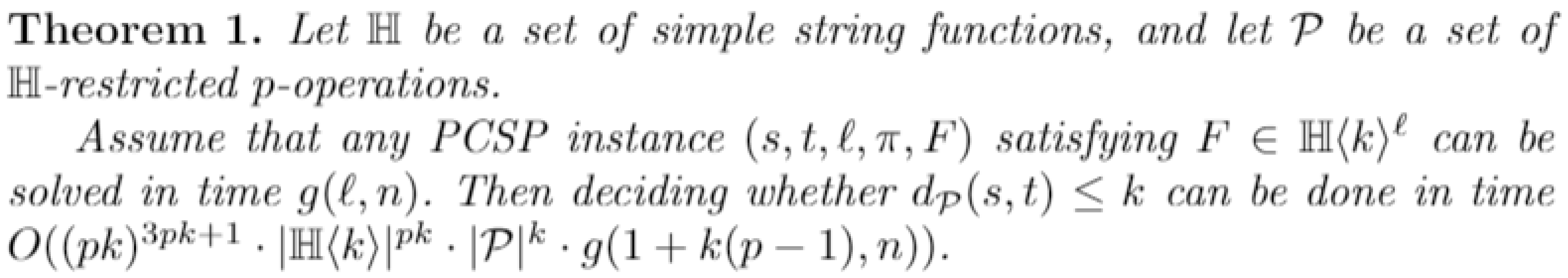
T = a a d c e b c d b



PCSP vs genomic distances

Theorem

If PCSP is FPT, then for **most** combination P of operations mentioned earlier, computing $dist_P(S,T)$ is FPT.



Theorem 1. *Let* $\mathbb{H}$ *be a set of simple string functions, and let* $\mathcal{P}$ *be a set of* $\mathbb{H}$*-restricted* p*-operations.*

Assume that any PCSP instance (s,t,ℓ,π,F) *satisfying* $F \in \mathbb{H}\langle k\rangle^{\ell}$ *can be solved in time* $g(\ell,n)$*. Then deciding whether* $d_{\mathcal{P}}(s,t) \leq k$ *can be done in time* $O((pk)^{3pk+1} \cdot |\mathbb{H}\langle k\rangle|^{pk} \cdot |\mathcal{P}|^{k} \cdot g(1+k(p-1),n))$.

PCSP vs genomic distances

Theorem

If PCSP is FPT, then for **most** combination P of operations mentioned earlier, computing $dist_P(S,T)$ is FPT.

- Idea : given S and T, to decide whether $dist_P(S,T) \leq k$:

```
For each $l = 1 .. k * c$  //for some constant c that depends on P
    For each permutation $\boldsymbol{\pi}$ of $[l]$
        If $S, T$ admit a partition into $l$ blocks that agrees with $\boldsymbol{\pi}$
            Compute $dist_P((1,2,\ldots,l),\boldsymbol{\pi})$
            If the distance is $k$ or less
                return true
return no
```

What about inversions?

$$S = a^+ \; b^+ \; \underline{c^+ \; d^+ \; a^-} \; d^+ \; b^+ \; c^-$$

$$\Downarrow$$

$$a^+ \; b^+ \; a^+ \; d^- \; \underline{c^- \; d^+ \; b^+ \; c^-}$$

$$\Downarrow$$

$$T = a^+ \; b^+ \; a^+ \; d^- \; c^+ \; b^- \; d^- \; c^+$$

What about inversions?

- Inversions => block partition in which some blocks are reversed, some not.

$$S = a^+\ b^+\ \underline{c^+\ d^+\ a^-}\ d^+\ b^+\ c^-$$

$$\Downarrow$$

$$a^+\ b^+\ a^+\ d^-\ \underline{c^-\ d^+\ b^+\ c^-}$$

$$\Downarrow$$

$$T = a^+\ b^+\ a^+\ d^-\ c^+\ b^-\ d^-\ c^+$$

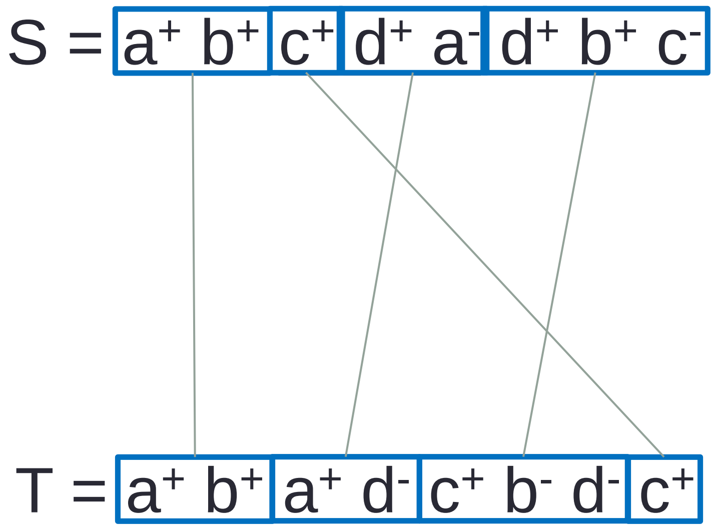
What about inversions?

- Inversions => block partition in which some blocks are reversed, some not.

$$\begin{tikzcd}
S = \boxed{a^+\ b^+}\boxed{c^+}\boxed{d^+\ a^-}\boxed{d^+\ b^+\ c^-} \arrow[d] \\
\boxed{a^+\ b^+}\boxed{a^+\ d^-}\boxed{c^-}\boxed{d^+\ b^+\ c^-} \arrow[d] \\
T = \boxed{a^+\ b^+}\boxed{a^+\ d^-}\boxed{c^+\ b^-\ d^-}\boxed{c^+}
\end{tikzcd}$$

What about inversions?

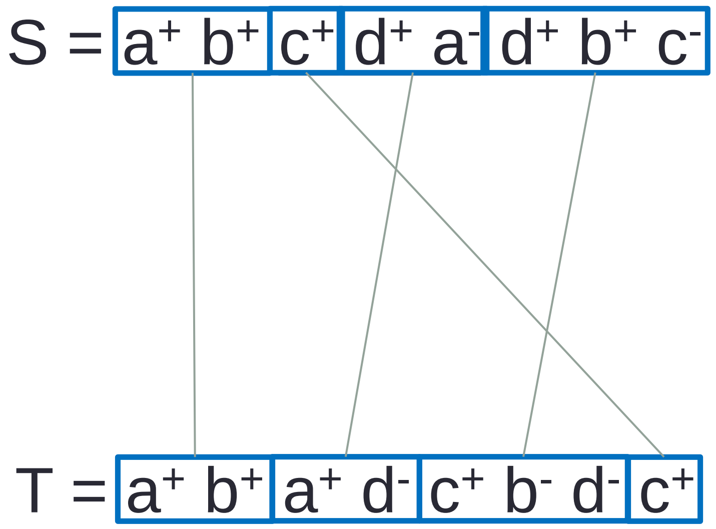
- Inversions => block partition in which some blocks are reversed, some not.



$S = a^+ b^+ | c^+ | d^+ a^- | d^+ b^+ c^-$
$T = a^+ b^+ | a^+ d^- | c^+ b^- d^- | c^+$

What about inversions?

- Inversions => block partition in which some blocks are reversed, some not.
- More generally, each block b_i might be affected by some string function $f_i(b_i)$.



Generalized PCSP

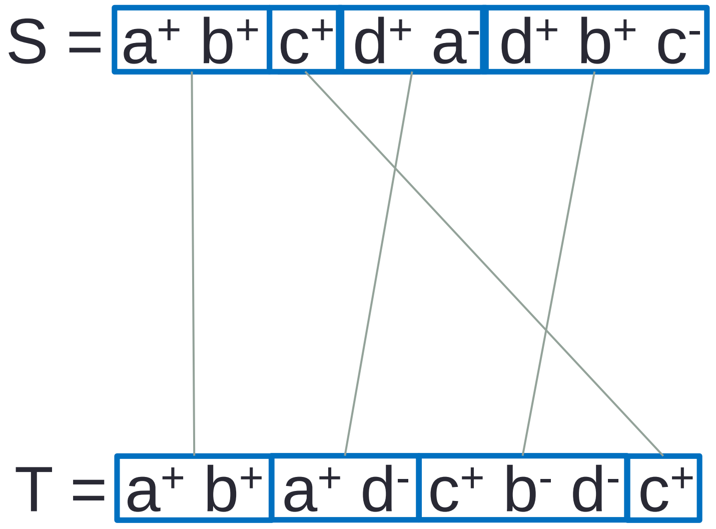
- Given : strings S and T, k, permutation $\boldsymbol{\pi}$, functions $f_1, \ldots, f_k$
- Find : a partition into k blocks of S and T, with a perfect matching that agrees with $\boldsymbol{\pi}$, such that block b_i is matched with block $f_i(b_i)$.

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

$S = a^+ b^+ c^+ d^+ a^- d^+ b^+ c^-$

$T = a^+ b^+ a^+ d^- c^+ b^- d^- c^+$

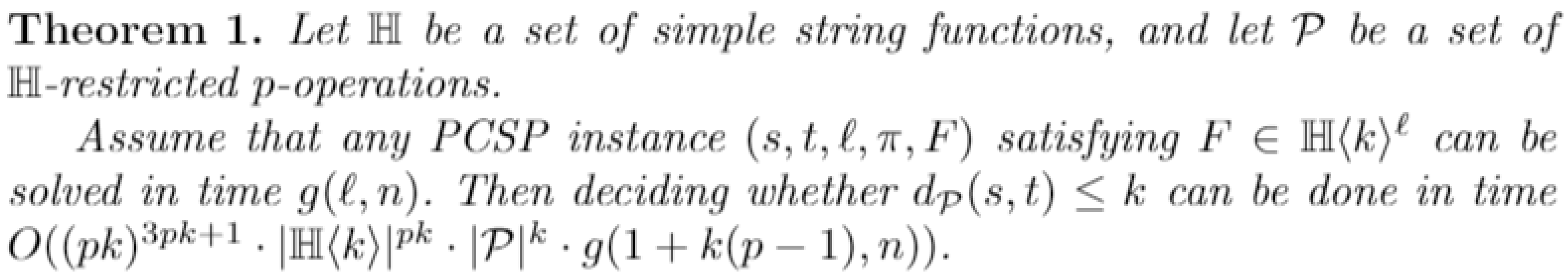


S = a+ b+ | c+ | d+ a- | d+ b+ c-
T = a+ b+ | a+ d- | c+ b- d- | c+

Generalized PCSP vs genomic distances

Theorem

If Generalized PCSP is FPT, then for **any** combination P of operations mentioned earlier, computing $dist_P(S,T)$ is FPT.



Theorem 1. *Let $\mathbb{H}$ be a set of simple string functions, and let $\mathcal{P}$ be a set of $\mathbb{H}$-restricted p-operations.*

Assume that any PCSP instance (s, t, ℓ, π, F) satisfying $F \in \mathbb{H}\langle k\rangle^{\ell}$ can be solved in time $g(\ell, n)$. Then deciding whether $d_{\mathcal{P}}(s,t) \leq k$ can be done in time $O((pk)^{3pk+1} \cdot |\mathbb{H}\langle k\rangle|^{pk} \cdot |\mathcal{P}|^{k} \cdot g(1 + k(p-1), n))$.

Generalized PCSP vs genomic distances

Theorem

If Generalized PCSP is FPT, then for **any** combination P of operations mentioned earlier, computing $dist_P(S,T)$ is FPT.

PROBLEM : PCSP is W[1]-hard in parameter k, generalized or not [Bulteau, Fellows, Komusiewicz, TBA]

We show that PCSP is FPT in $k + d$

Generalized PCSP vs genomic distances

Theorem

Generalized PCSP is FPT in $k + d$ and can be solved in time $O(d^{2k}(8k)^k n)$, if the f_i functions are in $\{id, rev, urev, flip\}$.

Reminder: $k =$ number of blocks

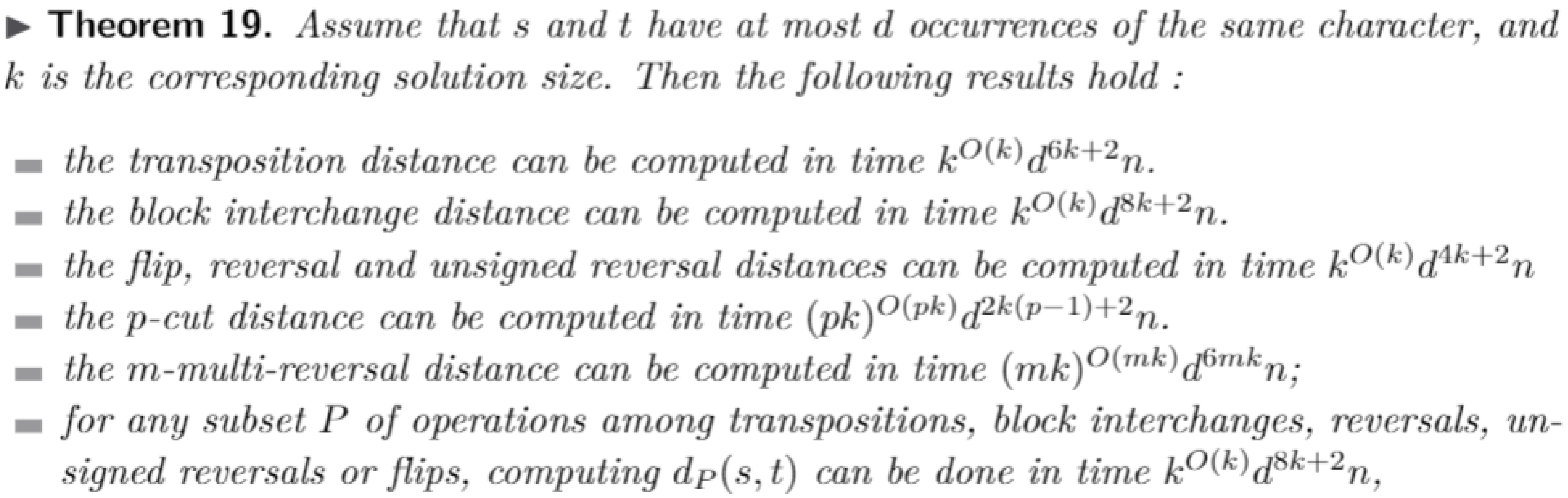
$d = \max \# \mathit{occurrences\ of\ a\ symbol}$

Generalized PCSP vs genomic distances

Theorem

Generalized PCSP is FPT in $k + d$ and can be solved in time $O(d^{2k}(8k)^k n)$, if the f_i functions are in $\{id, rev, urev, flip\}$.

Implications:



▶ **Theorem 19.** *Assume that s and t have at most d occurrences of the same character, and k is the corresponding solution size. Then the following results hold :*

- *the transposition distance can be computed in time $k^{O(k)}d^{6k+2}n$.*
- *the block interchange distance can be computed in time $k^{O(k)}d^{8k+2}n$.*
- *the flip, reversal and unsigned reversal distances can be computed in time $k^{O(k)}d^{4k+2}n$*
- *the p-cut distance can be computed in time $(pk)^{O(pk)}d^{2k(p-1)+2}n$.*
- *the m-multi-reversal distance can be computed in time $(mk)^{O(mk)}d^{6mk}n$;*
- *for any subset P of operations among transpositions, block interchanges, reversals, unsigned reversals or flips, computing $d_P(s,t)$ can be done in time $k^{O(k)}d^{8k+2}n$,*

Generalized PCSP vs genomic distances

Theorem

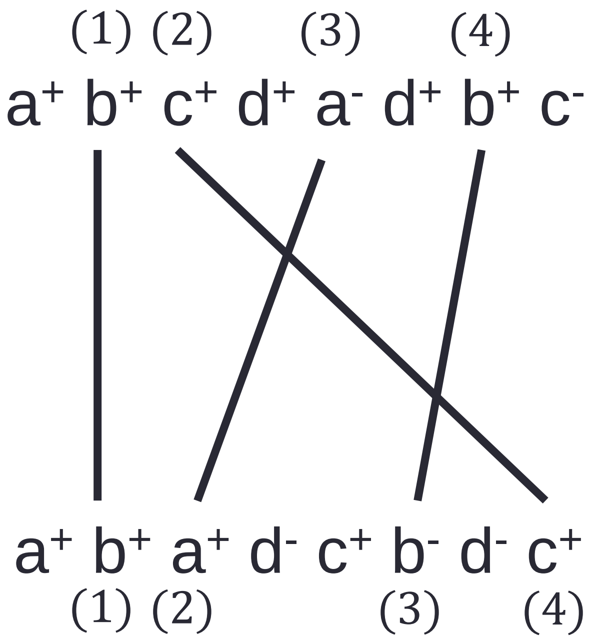
Generalized PCSP is FPT in $k + d$ and can be solved in time $O(d^{2k}(8k)^k n)$, if the f_i functions are in $\{id, rev, urev, flip\}$.

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Idea (from [Bulteau et al., WABI13]) :
Find **one pair** of matched characters **per block** of the solution.
Make sure that $\boldsymbol{\pi}$ and the f's are satisfied.


(1) (2) (3) (4)
$a^+ b^+ c^+ d^+ a^- d^+ b^+ c^-$
$a^+ b^+ a^+ d^- c^+ b^- d^- c^+$
(1) (2) (3) (4)

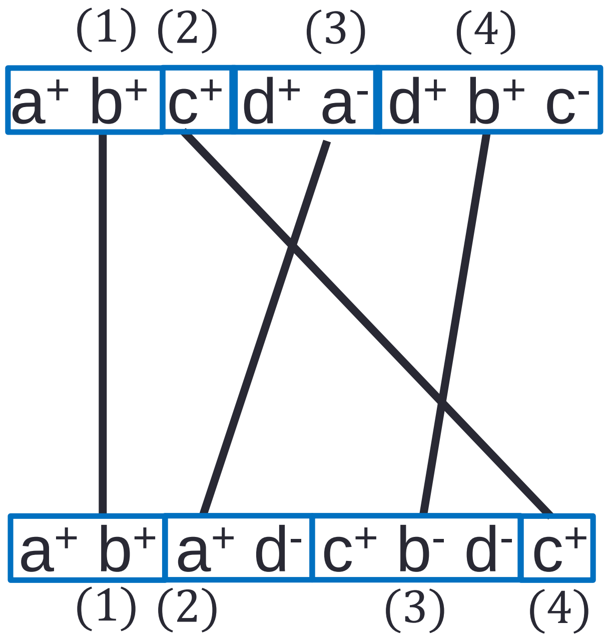
FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Idea (from [Bulteau et al., WABI13]) :
Find **one pair** of matched characters **per block** of the solution.
Make sure that $\boldsymbol{\pi}$ and the f's are satisfied.

If these were known, the entire blocks could be recovered.


(1) (2) (3) (4)
$a^+ b^+$ | c^+ | $d^+ a^-$ | $d^+ b^+ c^-$
$a^+ b^+$ | $a^+ d^-$ | $c^+ b^- d^-$ | c^+
(1) (2) (3) (4)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Initially, pick any character. Try to match with **every possible combination** of matching character + block number.

$a^+ \ b^+ \ c^+ \ d^+ \ a^- \ d^+ \ b^+ \ c^-$

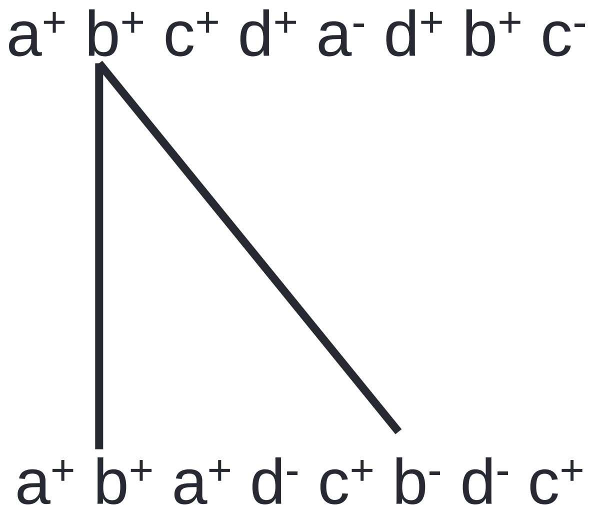
$a^+ \ b^+ \ a^+ \ d^- \ c^+ \ b^- \ d^- \ c^+$

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Initially, pick any character. Try to match with **every possible combination** of matching character + block number.



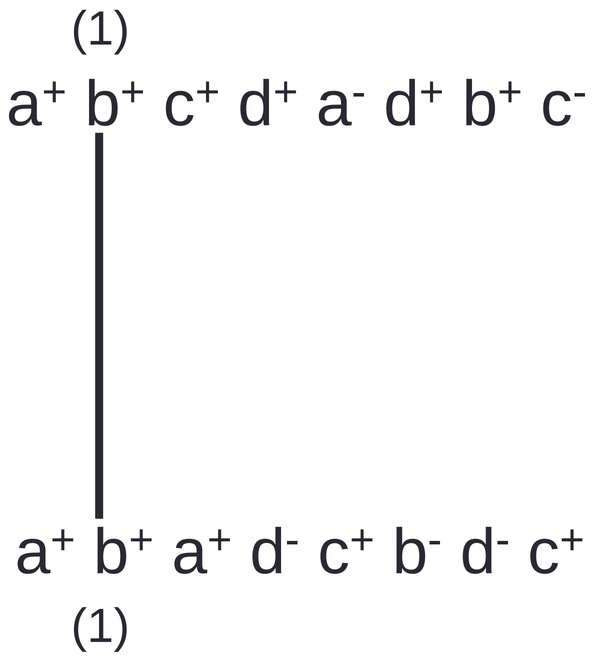
$a^+ \; b^+ \; c^+ \; d^+ \; a^- \; d^+ \; b^+ \; c^-$
$a^+ \; b^+ \; a^+ \; d^- \; c^+ \; b^- \; d^- \; c^+$

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Initially, pick any character. Try to match with **every possible combination** of matching character + block number.
Number of choices at most $2dk$

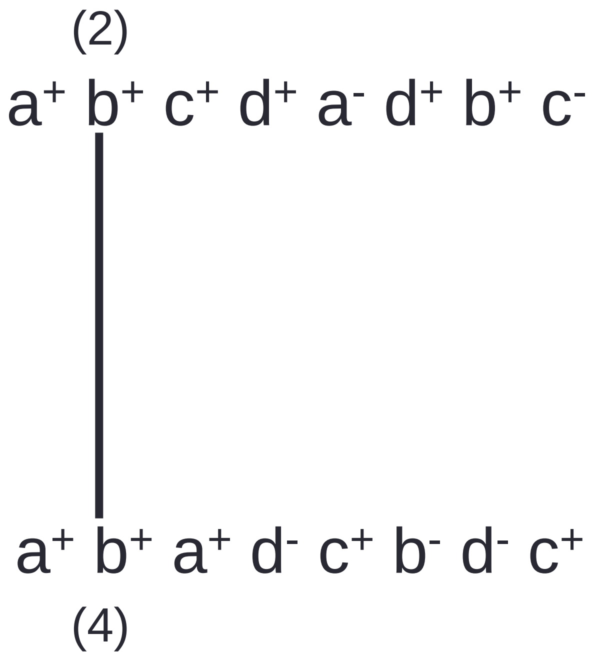


FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Initially, pick any character. Try to match with **every possible combination** of matching character + block number.
Number of choices at most $2dk$

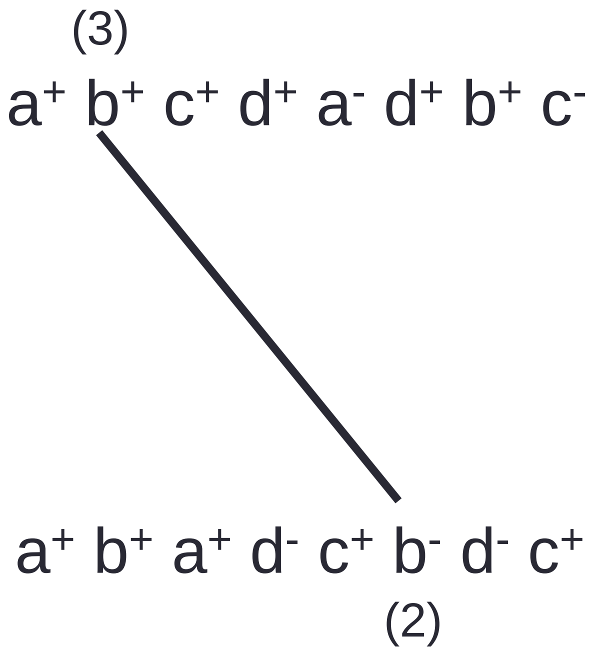

(2)
$a^+ \; b^+ \; c^+ \; d^+ \; a^- \; d^+ \; b^+ \; c^-$
$a^+ \; b^+ \; a^+ \; d^- \; c^+ \; b^- \; d^- \; c^+$
(4)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Initially, pick any character.
Try to match with **every possible combination** of matching character + block number.
Number of choices at most $2dk$

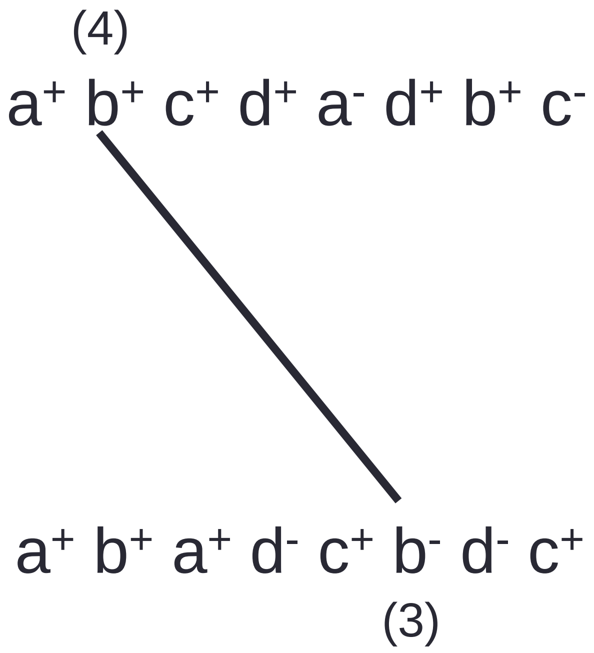


FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

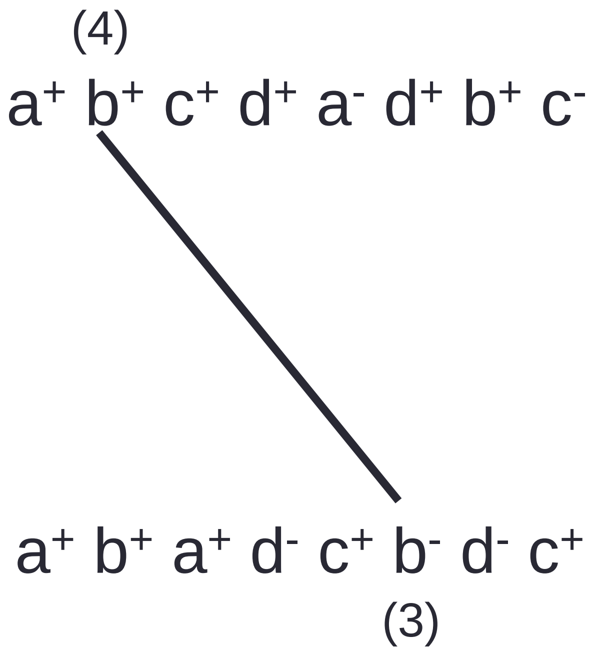
Initially, pick any character. Try to match with **every possible combination** of matching character + block number.
Number of choices at most $2dk$



FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\pi : (1, 4, 2, 3)$


(4)
$a^+ \; b^+ \; c^+ \; d^+ \; a^- \; d^+ \; b^+ \; c^-$
$a^+ \; b^+ \; a^+ \; d^- \; c^+ \; b^- \; d^- \; c^+$
(3)

Initially, pick any character. Try to match with **every possible combination** of matching character + block number.
Number of choices at most $2dk$

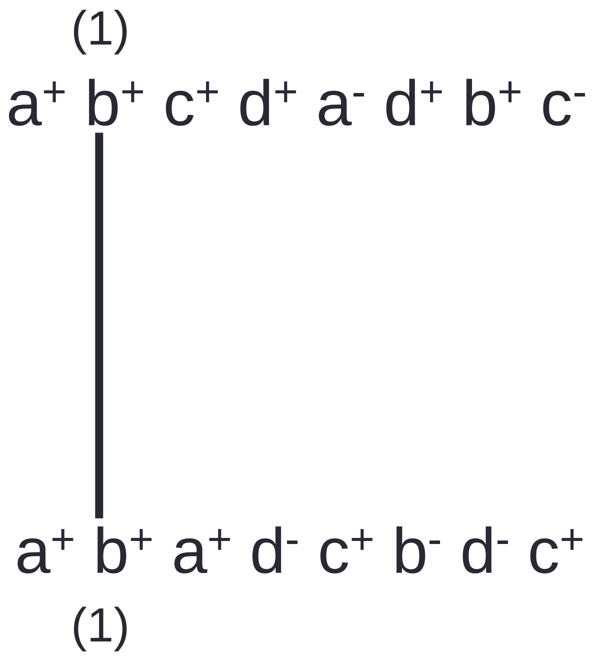
We call this a *fixed match*.

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Given a set of fixed matches, call two characters **matchable** if they could be in the same block as one of its closest fixed matches.

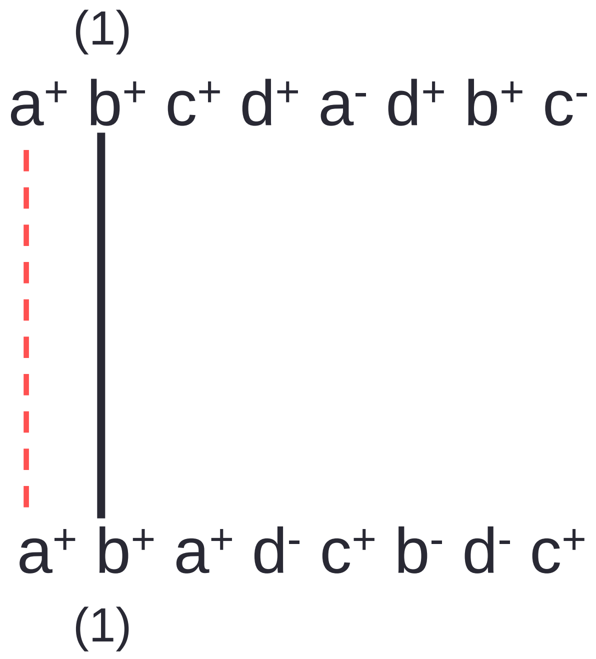

(1)
$a^+ b^+ c^+ d^+ a^- d^+ b^+ c^-$
$a^+ b^+ a^+ d^- c^+ b^- d^- c^+$
(1)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Given a set of fixed matches, call two characters **matchable** if they could be in the same block as one of its closest fixed matches.



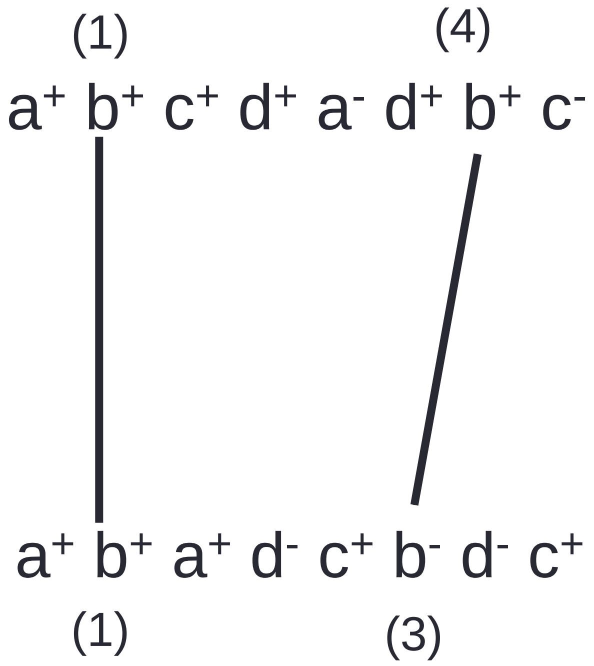
(1)
$a^+ b^+ c^+ d^+ a^- d^+ b^+ c^-$
$a^+ b^+ a^+ d^- c^+ b^- d^- c^+$
(1)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Given a set of fixed matches, call two characters **matchable** if they could be in the same block as one of its closest fixed matches.



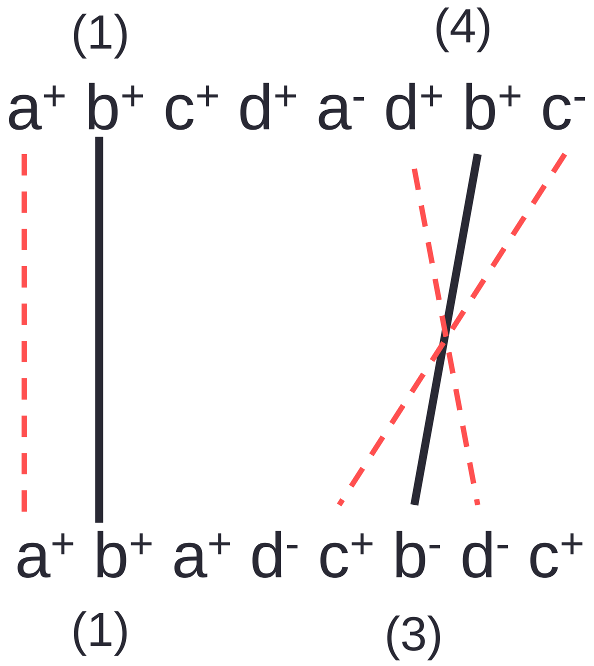
(1)
(4)
$a^+ \ b^+ \ c^+ \ d^+ \ a^- \ d^+ \ b^+ \ c^-$
$a^+ \ b^+ \ a^+ \ d^- \ c^+ \ b^- \ d^- \ c^+$
(1)
(3)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

Given a set of fixed matches, call two characters **matchable** if they could be in the same block as one of its closest fixed matches.

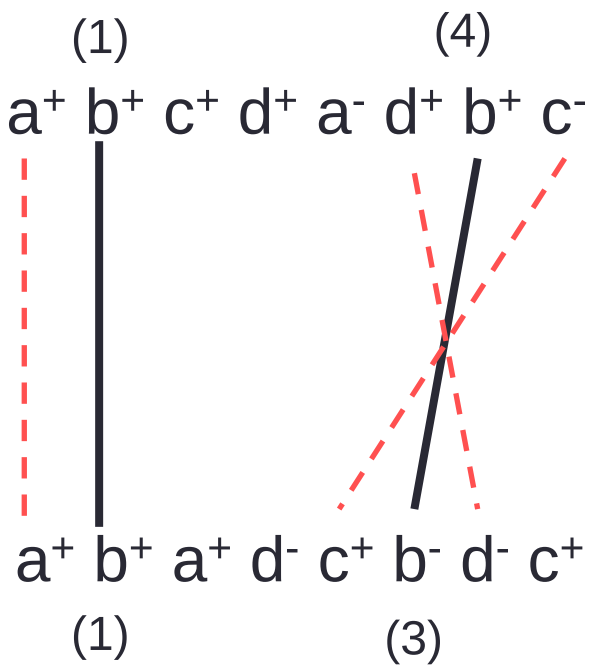


(1)
(4)
$a^+ \; b^+ \; c^+ \; d^+ \; a^- \; d^+ \; b^+ \; c^-$
$a^+ \; b^+ \; a^+ \; d^- \; c^+ \; b^- \; d^- \; c^+$
(1)
(3)

FPT algorithm for PCSP

$k = 4 \quad f_1 = f_2 = id, f_3 = f_4 = rev$

$\boldsymbol{\pi} : (1, 4, 2, 3)$

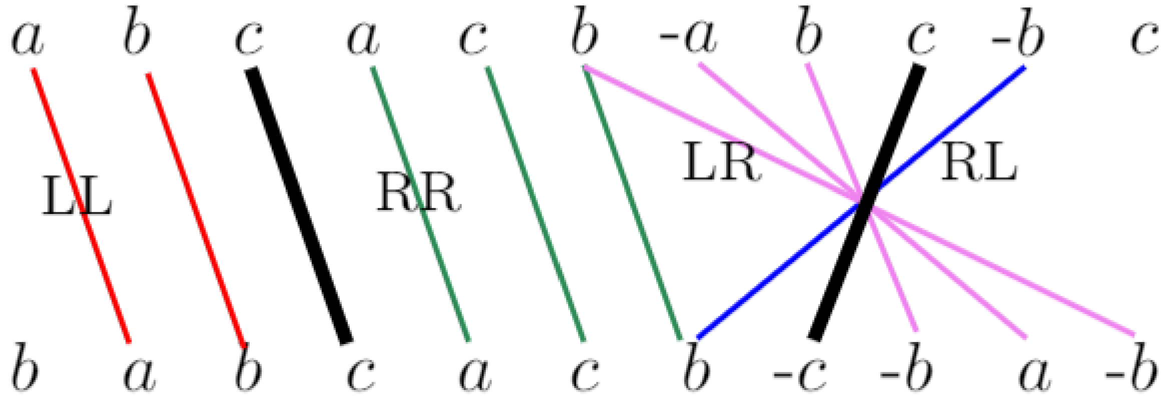


Given a set of fixed matches, call two characters **matchable** if they could be in the same block as one of its closest fixed matches.

As long as there is an unmatchable character, branch into all ways of matching it in a new fixed match.

FPT algorithm for PCSP

Match graph: vertices = characters, edge = fixed matches and matchable pairs



FPT algorithm for PCSP

A set of fixed matches is *complete* if each block of a solution corresponds to a distinct fixed match.

FPT algorithm for PCSP

A set of fixed matches is *complete* if each block of a solution corresponds to a distinct fixed match.

Lemma *[B et al] + generalized by us for f_i functions]*

If there is a path P with an odd number of vertices in the match graph, the set of fixed matches is not complete.

FPT algorithm for PCSP

A set of fixed matches is *complete* if each block of a solution corresponds to a distinct fixed match.

Lemma *[B et al. + generalized by us for f_i functions]*
If there is a path P with an odd number of vertices in the match graph, the set of fixed matches is not complete.

Hence, we some character that occurs in P must be added to a fixed match.

P has at most $4d$ characters, branch into $4d * 2dk = 8d^2k$ cases.

FPT algorithm for PCSP

Lemma

If the match graph has no odd path, then it is complete.

FPT algorithm for PCSP

Lemma

If the match graph has no odd path, then it is complete.

To summarize:

If there is an odd path P

Branch into $8d^2k$ ways to add a new block (i.e. a new fixed match)

Else, the graph is complete => there is a block partition

Since we need to create at most k blocks, create a recursion tree of depth at most k

```
function $PCSP(s, t, \ell, \pi, F, M)$
    //At the initial call, $M = \emptyset$
    if $M$ is not order-consistent then
        return "No solution"
    Construct $G(M)$
    if $G(M)$ has an odd path $(u_1, \ldots, u_h)$ then
        if $|M| = \ell$ then
            return "No solution"
        foreach $u_i$ on the path and each marker $v$ with the same symbol or negated
         symbol, such that $v$ is not already in a fixed-match of $M$ do
            foreach $b_i$ that is not already in a fixed-match of $M$ do
                Call $PCSP(s, t, \ell, \pi, F, M \cup \{(u, v, \pi(b_i), b_i)\})$
                if a positive answer was returned then
                    return "Yes"
            end
        end
        return "No solution"
    else
        return "Yes"
```

Generalized PCSP vs genomic distances

Theorem

Generalized PCSP is FPT in $k + d$ and can be solved in time $O(d^{2k}(8k)^k n)$, if the f_i functions are in $\{id, rev, urev, flip\}$.

Conclusion

- PCSP provides FPT algorithms for many string rearrangement problems.
- Allowable operations more generic than presented here, see paper.
- Most genomic problems are FPT in $k + d$
- Are they all FPT in k?
  - PCSP not useful here, because it is W[1]-hard in k…

- Genomic distances not handled by our framework : length-changing operations
  - E.g. block duplications or block deletions