Parallel algorithm for pattern matching problems under substring consistent equivalence relations

D. Jargalsaikhan, D. Hendrian, R. Yoshinaka, A. Shinohara

Tohoku University, Japan

CPM 2022

28 June 2022

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} \boxed{x_1 x_2 x_3 x_4 x_5 x_6 x_7} \\ \wr\wr \\ \boxed{y_1 y_2 y_3 y_4 y_5 y_6 y_7} \end{array}$$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} \boxed{x_1 x_2 x_3 x_4 x_5 x_6 x_7} \\ \wr\wr \\ \boxed{y_1 y_2 y_3 y_4 y_5 y_6 y_7} \end{array} \Longrightarrow \begin{array}{c} \boxed{x_3 x_4 x_5 x_6} \\ \wr\wr \\ \boxed{y_3 y_4 y_5 y_6} \end{array}$$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} \boxed{x_1 x_2 x_3 x_4 x_5 x_6 x_7} \\ \wr\wr \\ \boxed{y_1 y_2 y_3 y_4 y_5 y_6 y_7} \end{array} \implies \begin{array}{c} \boxed{x_3 x_4 x_5 x_6} \\ \wr\wr \\ \boxed{y_3 y_4 y_5 y_6} \end{array}$$

- Parameterized pattern matching (matching with bijection) [Baker1996]:
  Find $f(P)$ in T with an arbitrary bijection f over Σ

$P =$ `abac`

$T =$ `ac`$\boxed{\texttt{abac}}$`acb`$\boxed{\texttt{abac}}$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \le i \le j \le |X|.$$

$$\begin{array}{c} x_1 x_2 \boxed{x_3 x_4 x_5 x_6} x_7 \\ \wr\wr \\ y_1 y_2 \boxed{y_3 y_4 y_5 y_6} y_7 \end{array} \implies \begin{array}{c} x_3 x_4 x_5 x_6 \\ \wr\wr \\ y_3 y_4 y_5 y_6 \end{array}$$

- Parameterized pattern matching (matching with bijection) [Baker1996]:
  Find $f(P)$ in T with an arbitrary bijection f over Σ

$P = \texttt{abac} \approx \texttt{acab}$

$T = \texttt{acabacacbabac}$

$f(\texttt{a}) = \texttt{a}$, $f(\texttt{b}) = \texttt{c}$, $f(\texttt{c}) = \texttt{b}$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} x_1 x_2 \boxed{x_3 x_4 x_5 x_6} x_7 \\ \wr\wr \\ y_1 y_2 \boxed{y_3 y_4 y_5 y_6} y_7 \end{array} \Longrightarrow \begin{array}{c} \boxed{x_3 x_4 x_5 x_6} \\ \wr\wr \\ \boxed{y_3 y_4 y_5 y_6} \end{array}$$

- Parameterized pattern matching (matching with bijection) [Baker1996]:
  Find $f(P)$ in T with an arbitrary bijection f over Σ

$P = \mathtt{abac} \approx \mathtt{acab} \approx \mathtt{cacb}$

$T = \mathtt{acabacacbabac}$

$f(\mathrm{a}) = \mathrm{a},\ f(\mathrm{b}) = \mathrm{c},\ f(\mathrm{c}) = \mathrm{b}$

$f(\mathrm{a}) = \mathrm{c},\ f(\mathrm{b}) = \mathrm{a},\ f(\mathrm{c}) = \mathrm{b}$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

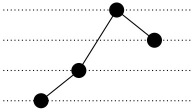
- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} \boxed{x_1 x_2 x_3 x_4 x_5 x_6 x_7} \\ \wr\wr \\ \boxed{y_1 y_2 y_3 y_4 y_5 y_6 y_7} \end{array} \quad \Longrightarrow \quad \begin{array}{c} \boxed{x_3 x_4 x_5 x_6} \\ \wr\wr \\ \boxed{y_3 y_4 y_5 y_6} \end{array}$$

- Order-isomorphic (order-preserving) matching [Kubica+2013,Kim+2014]:
  (Σ is linearly ordered)

$$1\ 2\ 4\ 3 \approx 2\ 4\ 9\ 5 \not\approx 1\ 2\ 3\ 4$$



Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

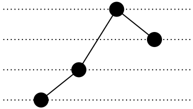
- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \le i \le j \le |X|.$$

$$\begin{array}{c} x_1x_2\boxed{x_3x_4x_5x_6}x_7 \\ \wr\wr \\ y_1y_2\boxed{y_3y_4y_5y_6}y_7 \end{array} \Longrightarrow \begin{array}{c} \boxed{x_3x_4x_5x_6} \\ \wr\wr \\ \boxed{y_3y_4y_5y_6} \end{array}$$

- Order-isomorphic (order-preserving) matching [Kubica+2013,Kim+2014]:
  (Σ is linearly ordered)

$$1\ 2\ 4\ 3 \approx 2\ 4\ 9\ 5 \not\approx 1\ 2\ 3\ 4$$



$$P = 1\ 2\ 4\ 3$$
$$T = 1\ 2\ 4\ 9\ 5\ 6\ 8\ 7$$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

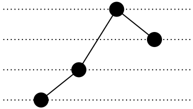
- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \le i \le j \le |X|.$$

$$\begin{array}{c} x_1 x_2 x_3 x_4 x_5 x_6 x_7 \\ \wr\wr \\ y_1 y_2 y_3 y_4 y_5 y_6 y_7 \end{array} \Longrightarrow \begin{array}{c} x_3 x_4 x_5 x_6 \\ \wr\wr \\ y_3 y_4 y_5 y_6 \end{array}$$

- Order-isomorphic (order-preserving) matching [Kubica+2013,Kim+2014]:
  (Σ is linearly ordered)

$$1\ 2\ 4\ 3 \approx 2\ 4\ 9\ 5 \not\approx 1\ 2\ 3\ 4$$



$$P = 1\ 2\ 4\ 3$$
$$T = 1\ 2\ 4\ 9\ 5\ 6\ 8\ 7$$

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \leq i \leq j \leq |X|.$$

$$\begin{array}{c} \boxed{x_1x_2x_3x_4x_5x_6x_7} \\ \wr \\ \boxed{y_1y_2y_3y_4y_5y_6y_7} \end{array} \quad \Longrightarrow \quad \begin{array}{c} \boxed{x_3x_4x_5x_6} \\ \wr \\ \boxed{y_3y_4y_5y_6} \end{array}$$

- Exact matching, parameterized pattern matching, order-isomorphic matching, Cartesian-tree matching, etc.

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \le i \le j \le |X|.$$

$$\begin{array}{c} \boxed{x_1 x_2 x_3 x_4 x_5 x_6 x_7} \\ \wr\wr \\ \boxed{y_1 y_2 y_3 y_4 y_5 y_6 y_7} \end{array} \quad \Longrightarrow \quad \begin{array}{c} \boxed{x_3 x_4 x_5 x_6} \\ \wr\wr \\ \boxed{y_3 y_4 y_5 y_6} \end{array}$$

- Exact matching, parameterized pattern matching, order-isomorphic matching, Cartesian-tree matching, etc.
- KMP-type algorithm framework [Matsuoka+ 2016]
  - $\mathrm{O}(\tau_{\approx}(n, m) + \xi_{\approx}(n, m) \cdot (n + m))$ time
    where $\tau_{\approx}$ and $\xi_{\approx}$ depend on the concerned SCER.
    (they are often very small.)

Substring consistent equivalence relations (SCERs) [Matsuoka et al. 2016]

- Equivalence $\approx$ on strings is *substring-consistent* if

$$X \approx Y \text{ implies } |X| = |Y| \text{ and } X[i:j] \approx Y[i:j] \text{ for } 1 \le i \le j \le |X|.$$

$$\begin{array}{c} \boxed{x_1x_2\boxed{x_3x_4x_5x_6}x_7} \\ \wr\wr \\ \boxed{y_1y_2\boxed{y_3y_4y_5y_6}y_7} \end{array} \Longrightarrow \begin{array}{c} \boxed{x_3x_4x_5x_6} \\ \wr\wr \\ \boxed{y_3y_4y_5y_6} \end{array}$$

⋆ Remark

$$\begin{array}{cc} \boxed{x_1x_2x_3x_4} & \boxed{x_5x_6x_7} \\ \wr\wr & \wr\wr \\ \boxed{y_1y_2y_3y_4} & \boxed{y_5y_6y_7} \end{array} \not\Longrightarrow \begin{array}{c} \boxed{x_1x_2x_3x_4x_5x_6x_7} \\ \wr\wr \\ \boxed{y_1y_2y_3y_4y_5y_6y_7} \end{array}$$

E.g., parameterized matching:
ab $\approx$ ab and ab $\approx$ ac but abab $\not\approx$ abac.

Our contribution

- Parallel algorithm framework for SCER-matching
  - $\mathrm{O}(\tau_n^{\mathrm{t}} + \xi_m^{\mathrm{t}} \cdot \log^3 m)$ time
    $\mathrm{O}(\tau_n^{\mathrm{w}} + \xi_m^{\mathrm{w}} \cdot n \log^2 m)$ work, where
    - n: text length
    - m: pattern length
    - parameters depending on an SCER.

	τ_n^{t}	τ_n^{w}	ξ_m^{t}	ξ_m^{w}
Exact	$\mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{O}(1)$
Parametererized	$\mathrm{O}(\log n)$	$\mathrm{O}(n \log n)$	$\mathrm{O}(1)$	$\mathrm{O}(1)$
Cartesian-tree	$\mathrm{O}(\log n)$	$\mathrm{O}(n \log n)$	$\mathrm{O}(\log m)$	$\mathrm{O}(m \log m)$
...				

⋆ Parallel computation model:
Priority Concurrent Read Concurrent Write Parallel Random-Access Machine (P-CRCW PRAM)

- Multiple processors can read the same memory at the same time,
- In case multiple processors simultaneously try to write,
  only the processor with the smallest index succeeds.

Our contribution

- Parallel algorithm framework for SCER-matching
  - $O(\tau_n^{\mathrm{t}} + \xi_m^{\mathrm{t}} \cdot \log^3 m)$ time
    $O(\tau_n^{\mathrm{w}} + \xi_m^{\mathrm{w}} \cdot n \log^2 m)$ work, where
    - n: text length
    - m: pattern length
    - parameters depending on an SCER.

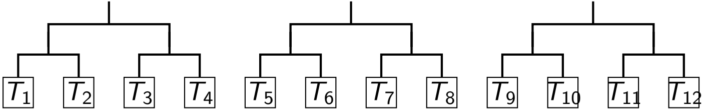
	τ_n^{t}	τ_n^{w}	ξ_m^{t}	ξ_m^{w}
Exact	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Parametererized	$O(\log n)$	$O(n \log n)$	$O(1)$	$O(1)$
Cartesian-tree	$O(\log n)$	$O(n \log n)$	$O(\log m)$	$O(m \log m)$
...				

⋆ Parallel computation model:
Priority Concurrent Read Concurrent Write Parallel Random-Access Machine (P-CRCW PRAM)
- Multiple processors can read the same memory at the same time,
- In case multiple processors simultaneously try to write,
  only the processor with the smallest index succeeds.

Duel & sweep

Parallel algorithm for exact matching by Vishkin (1985)

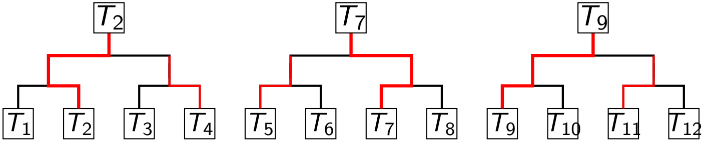
1. [Duel] *Candidates*, $T_i = T[i : i + m - 1]$ with $m = |P|$, *duel* each other repeatedly.
   $\rightarrow$ All occurrences and some non-occurrences survive the duels.



Duel & sweep

Parallel algorithm for exact matching by Vishkin (1985)

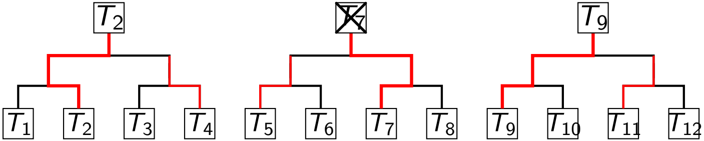
1. [Duel] *Candidates*, $T_i = T[i : i + m - 1]$ with $m = |P|$, *duel* each other repeatedly.
   $\rightarrow$ All occurrences and some non-occurrences survive the duels.



Duel & sweep

Parallel algorithm for exact matching by Vishkin (1985)

1. [Duel] *Candidates*, $T_i = T[i : i + m - 1]$ with $m = |P|$, *duel* each other repeatedly.
   $\rightarrow$ All occurrences and some non-occurrences survive the duels.
2. [Sweep] Kill the remaining non-occurrences.



Duel & sweep

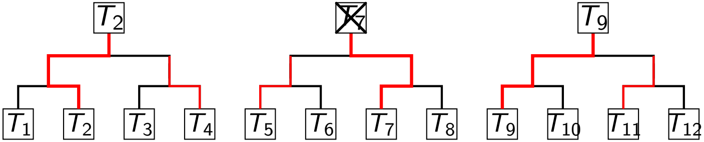
Parallel algorithm for exact matching by Vishkin (1985)

1. [Duel] *Candidates*, $T_i = T[i : i + m - 1]$ with $m = |P|$, *duel* each other repeatedly.
   $\rightarrow$ All occurrences and some non-occurrences survive the duels.
2. [Sweep] Kill the remaining non-occurrences.

The idea is applied to

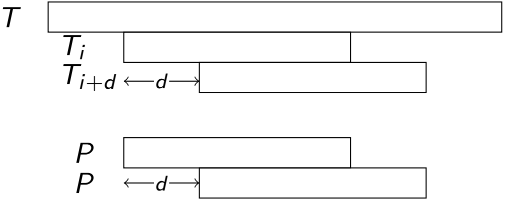
- two-dimensional exact matching (serial) [Amir+ 1994]
- two-dimensional parameterized matching (serial) [Cole+ 2014]
- order-isomorphic matching (serial/parallel) [Jargalsaikhan+ 2018][Jargalsaikhan 2022]

  (Our parallel algorithm presented at SOFSEM 2020 was in error...)



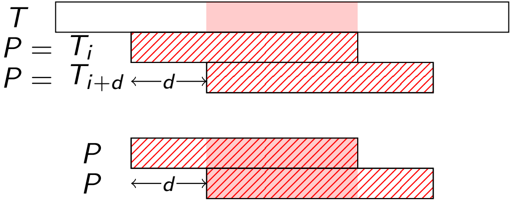
Dueling in exact matching

- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* ($d \leq m/2$)



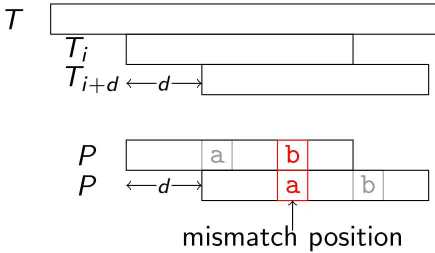
Dueling in exact matching

- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* ($d \leq m/2$)



Dueling in exact matching

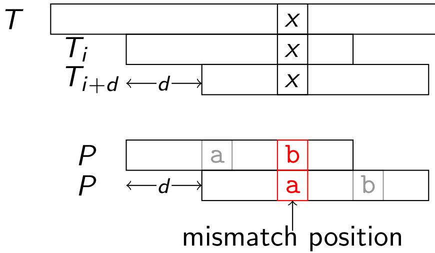
- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* $(d \leq m/2)$



- Assume a mismatch when P is superimposed on itself with offset d.
- Overlapped candidates of distance d cannot be occurrences simultaneously.
- By checking a single position, either T_i or T_{i+d} can be eliminated.

Dueling in exact matching

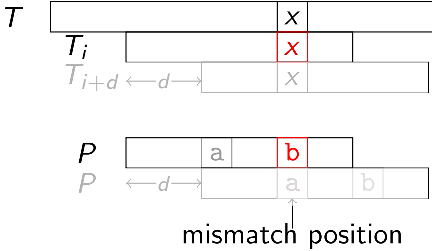
- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* ($d \leq m/2$)



- Assume a mismatch when P is superimposed on itself with offset d.
- Overlapped candidates of distance d cannot be occurrences simultaneously.
- By checking a single position, either T_i or T_{i+d} can be eliminated.

Dueling in exact matching

- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* ($d \leq m/2$)

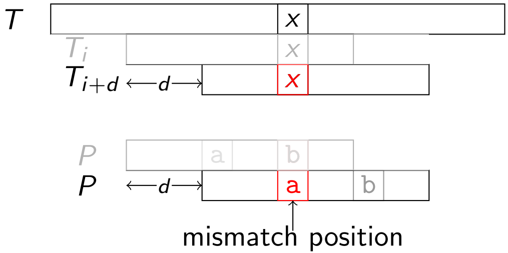


If $x = \texttt{a}$, T_i cannot be an occurrence.

- Assume a mismatch when P is superimposed on itself with offset d.
- Overlapped candidates of distance d cannot be occurrences simultaneously.
- By checking a single position, either T_i or T_{i+d} can be eliminated.

Dueling in exact matching

- Candidate: each $T_i = T[i : i + m - 1]$
  - (precisely, a candidate is a position i rather than a string)
- Largely overlapping candidates T_i and T_{i+d} *duel!* ($d \leq m/2$)



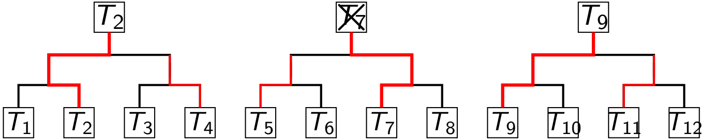
If $x = \text{a}$, T_i cannot be an occurrence.
If $x \neq \text{a}$, T_{i+d} cannot be an occurrence.

- Assume a mismatch when P is superimposed on itself with offset d.
- Overlapped candidates of distance d cannot be occurrences simultaneously.
- By checking a single position, either T_i or T_{i+d} can be eliminated.

Duel & sweep

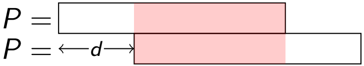
0. Preprocessing the pattern for determining the manner of the dueling.
    - For each offset d, find a mismatch position $W[d]$.
1. [Duel] (Largely) overlapping candidates *duel* each other repeatedly.
    - $\lfloor \log_2 m \rfloor$ rounds of parellel duels
2. [Sweep] Kill the remaining non-occurrences.
    - Survivor candidates are sparse enough to validate naively.

$\lfloor \log_2 m \rfloor$ rounds
of parallel duels



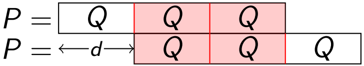
Remark

- What if there is no mismatch of P for offset $d \leq m/2$?



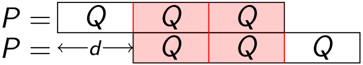
Remark

- What if there is no mismatch of P for offset $d \leq m/2$?
  $\implies P = Q^k$ for some Q of length d. (P is *periodic*.)
- Run the duel & sweep algorithm for *aperiodic* Q and T.
- k consecutive occurrences of Q form an occurrence of P.



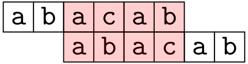
Remark

- What if there is no mismatch of P for offset $d \leq m/2$?
  $\implies P = Q^k$ for some Q of length d. (P is *periodic*.)
- Run the duel & sweep algorithm for *aperiodic* Q and T.
- k consecutive occurrences of Q form an occurrence of P.



⋆ This is not necessarily the case for SCERs.

- E.g. Parameterized matching
  $P = \mathtt{abacab}$. No mismatch position for offset 2, but $P \not\approx (\mathtt{ab})^3$.



$\mathtt{acab} \approx \mathtt{abac}$

- $ab \approx ab$ and $ab \approx ac$ but $abab \not\approx abac$.

SCER-encoding

Lemma (Amir and Kondratovsky, CPM 2019)

Every SCER $\approx$ *admits* $\phi : \Sigma^* \to \Delta^*$ *such that*

- $|X| = |\phi(X)|$
- $X \approx Y$ *iff* $\phi(X) = \phi(Y)$
- $\phi(X)[1 : i] = \phi(X[1 : i])$
- $\phi(X)[i] = \phi(Y)[i]$ *implies* $\phi(X[j+1 : k])[i-j] = \phi(Y[j+1 : k])[i-j]$

- Reducing SCER-matchings to exact matching (in a limited way)

SCER-encoding

Lemma (Amir and Kondratovsky, CPM 2019)

Every SCER $\approx$ *admits* $\phi : \Sigma^* \to \Delta^*$ *such that*

- $|X| = |\phi(X)|$
- $X \approx Y$ *iff* $\phi(X) = \phi(Y)$
- $\phi(X)[1:i] = \phi(X[1:i])$
- $\phi(X)[i] = \phi(Y)[i]$ *implies* $\phi(X[j+1:k])[i-j] = \phi(Y[j+1:k])[i-j]$

- Reducing SCER-matchings to exact matching (in a limited way)

Example: prev-encoding for parameterize matching

Prev-encoding $\mathrm{prev}(\cdot) : \Sigma^* \to \mathbb{N}^*$ for parameterized matching:

$$\mathrm{prev}(\mathtt{a\,b\,a\,c\,a\,b}) = 0\,0\,2\,0\,2\,4 = \mathrm{prev}(\mathtt{x\,y\,x\,z\,x\,y})$$

- ⋆ Each number indicates the distance to the previous occurrence of the same letter.

Example: prev-encoding for parameterize matching

Prev-encoding $\mathrm{prev}(\cdot) : \Sigma^* \to \mathbb{N}^*$ for parameterized matching:

$$\mathrm{prev}(\overset{0}{\mathtt{a}}\,\underset{0}{\mathtt{b}}\,\overset{2}{\mathtt{a}}\,\underset{0}{\mathtt{c}}\,\overset{2}{\mathtt{a}}\,\underset{4}{\mathtt{b}}) = 0\,0\,2\,0\,2\,4 = \mathrm{prev}(\mathtt{x\,y\,x\,z\,x\,y})$$

- ⋆ Each number indicates the distance to the previous occurrence of the same letter.

Example: $P = \mathtt{xyxz}$ in $T = \mathtt{abacab}$.

$$\mathrm{prev}(\mathtt{abacab}) = 002024$$
$$\mathrm{prev}(\mathtt{xyxz}) = 0020$$

Example: prev-encoding for parameterize matching

Prev-encoding $\mathrm{prev}(\cdot) : \Sigma^* \to \mathbb{N}^*$ for parameterized matching:

$$\mathrm{prev}(\mathtt{a\,b\,a\,c\,a\,b}) = 0\,0\,2\,0\,2\,4 = \mathrm{prev}(\mathtt{x\,y\,x\,z\,x\,y})$$

⋆ Each number indicates the distance to the previous occurrence of the same letter.

Example: $P = \mathtt{xyxz}$ in $T = \mathtt{abacab}$.

$$\mathrm{prev}(\mathtt{abacab}) = 002024$$
$$\mathrm{prev}(\mathtt{xyxz}) = 0020 \quad = \mathrm{prev}(\mathtt{acab})$$

For finding $\mathtt{xyxz} \approx \mathtt{acab}$ in $\mathtt{abacab}$, we need to re-encode text substrings.

- $\tau^{\mathrm{t}}(n)$, $\tau^{\mathrm{w}}(n)$: for encoding a whole string of length n,
- $\xi^{\mathrm{t}}(m)$, $\xi^{\mathrm{w}}(m)$: for re-encoding a single element of an encoded string of length m.

Outline of our algorithm

0. Pattern is preprocessed for determining the manner of the dueling.
1. [Duel] Candidates T_i duel each other repeatedly.
2. [Sweep] Kill the remaining non-occurrences.

Outline of our algorithm

We cannot assume that P is *aperiodic*.

0. Pattern is preprocessed for determining the manner of the dueling.
   - *Some offsets d may have no mismatching positions.*
1. [Duel] Candidates T_i duel each other repeatedly.
   - *Some pairs of candidates cannot perform duels.*
2. [Sweep] Kill the remaining non-occurrences.
   - *Survivors are not necessarily few.*

Outline of our algorithm

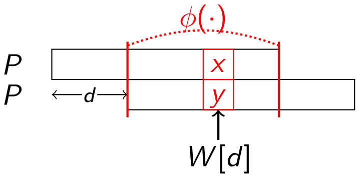
We cannot assume that P is *aperiodic*.

0. Pattern is preprocessed for determining the manner of the dueling.
   - *Some offsets d may have no mismatching positions.*
1. [Duel] Candidates T_i duel each other repeatedly.
   - *Some pairs of candidates cannot perform duels.*
   - *But survivors satisfy a good property for efficient sweep.*
2. [Sweep] Kill the remaining non-occurrences.
   - *Survivors are not necessarily few.*

Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

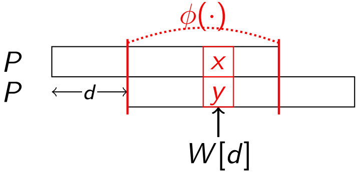
- $W[d] = 0$ iff $P[1:m-d] \approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1:m-d])[k] \neq \phi(P[d+1:m])[k]$.



Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

- $W[d] = 0$ iff $P[1:m-d] \approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1:m-d])[k] \neq \phi(P[d+1:m])[k]$.



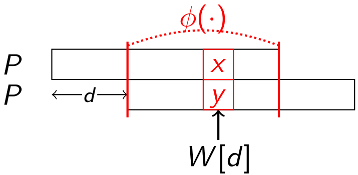

```
a b a c a b
  a b a c a b
    a b a c a b
      a b a c a b
        a b a c a b
          a b a c a b
```

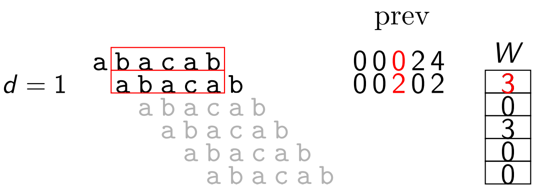
W
3
0
3
0
0

Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

- $W[d] = 0$ iff $P[1:m-d] \approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1:m-d])[k] \neq \phi(P[d+1:m])[k]$.

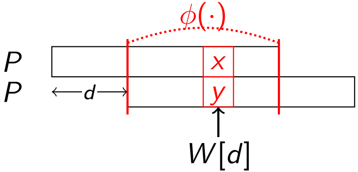


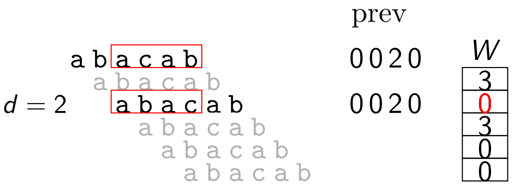


Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1 : m - d])$ and $\phi(P[d + 1 : m])$

- $W[d] = 0$ iff $P[1 : m - d] \approx P[d + 1 : m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1 : m - d])[k] \neq \phi(P[d + 1 : m])[k]$.

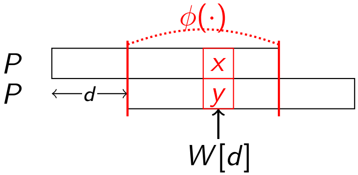


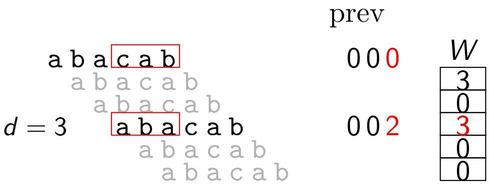


Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

- $W[d] = 0$ iff $P[1:m-d] \approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1:m-d])[k] \neq \phi(P[d+1:m])[k]$.

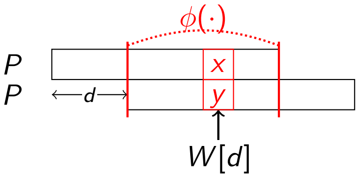


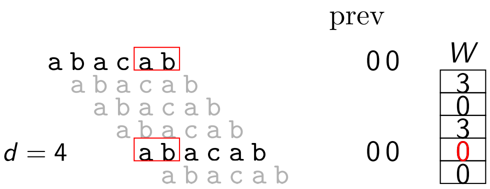


Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

- $W[d]=0$ iff $P[1:m-d]\approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k>0$ s.t. $\phi(P[1:m-d])[k]\neq\phi(P[d+1:m])[k]$.

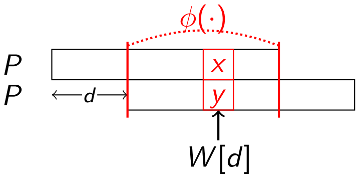


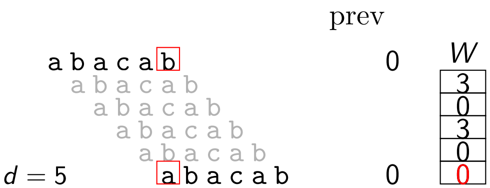


Preprocess – Witness table W of P determines the manner of dueling

$W[d]$ is a mismatch position of $\phi(P[1:m-d])$ and $\phi(P[d+1:m])$

- $W[d] = 0$ iff $P[1:m-d] \approx P[d+1:m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1:m-d])[k] \neq \phi(P[d+1:m])[k]$.

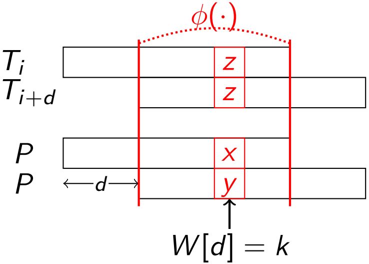




Preprocess – Witness table W of P determines manner of dueling

$W[d]$ is a mismatch position of $P[1 : m - d]$ and $P[d + 1 : m]$ (under ϕ)

- $W[d] = 0$ iff $P[1 : m - d] \approx P[d + 1 : m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1 : m - d])[k] \neq \phi(P[d + 1 : m])[k]$.



Lemma

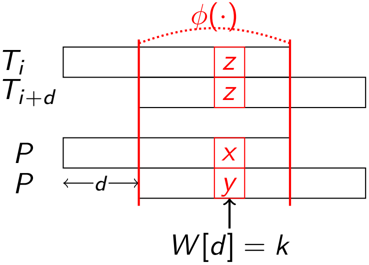
Suppose $W[d] \neq 0$.

- *If $\phi(T_{i+d})[k] = \phi(P)[k]$, then $T_i \not\approx P$.*
- *If $\phi(T_{i+d})[k] \neq \phi(P)[k]$, then $T_{i+d} \not\approx P$.*

Preprocess – Witness table W of P determines manner of dueling

$W[d]$ is a mismatch position of $P[1 : m-d]$ and $P[d+1 : m]$ (under ϕ)

- $W[d] = 0$ iff $P[1 : m-d] \approx P[d+1 : m]$;
- Otherwise, $W[d]$ is some position $k > 0$ s.t. $\phi(P[1 : m-d])[k] \neq \phi(P[d+1 : m])[k]$.



Lemma

Suppose $W[d] \neq 0$.

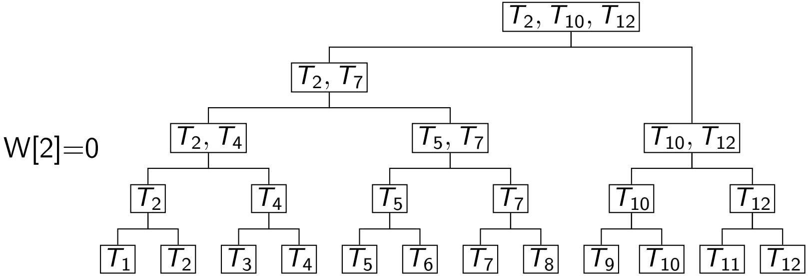
- *If $\phi(T_{i+d})[k] = \phi(P)[k]$, then $T_i \not\approx P$.*
- *If $\phi(T_{i+d})[k] \neq \phi(P)[k]$, then $T_{i+d} \not\approx P$.*

Lemma

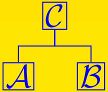
A witness table can be computed by $\mathrm{O}(\tau_m^{\mathrm{t}} + \xi_m^{\mathrm{t}} \log^2 m)$ time and $\mathrm{O}(\tau_m^{\mathrm{w}} + \xi_m^{\mathrm{w}} m \log^2 m)$ work on the P-CRCW PRAM.

Dueling stage

- Condidates T_i and T_{i+d} duel each other repeatedly, *if possiable* $(W[d] \neq 0)$.
- Otherwise, they are *consistent* $(W[d] = 0$ or $d \geq m)$.
  - A candidate set is *consistent* if every pair from the set is consistent.
- Survivors will be consistent. $\rightarrow$ Sweeping stage takes advantage of the consistency



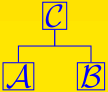
Merging consistent candidate bags



Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.

Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

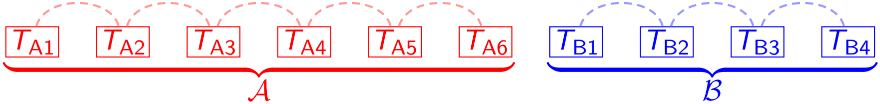
Merging consistent candidate bags



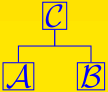
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.
Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.



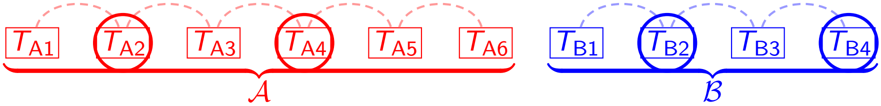
Merging consistent candidate bags



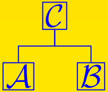
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.
Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.



Merging consistent candidate bags



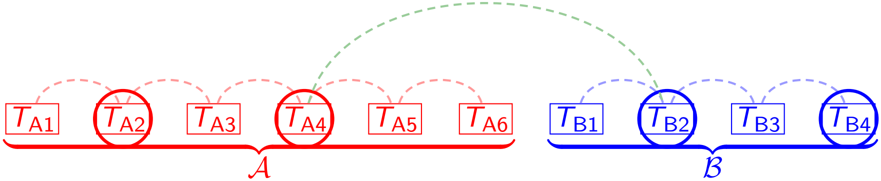
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.

Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$

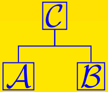
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.



Merging consistent candidate bags



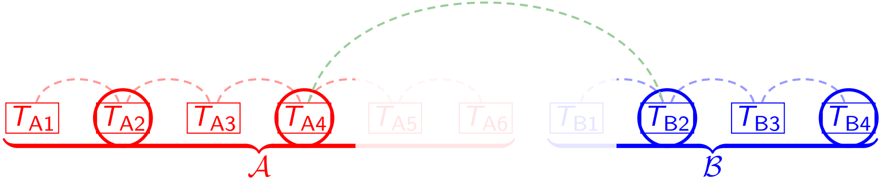
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.

Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$

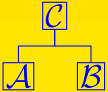
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.



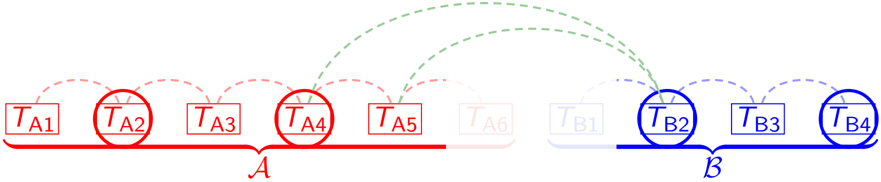
Merging consistent candidate bags



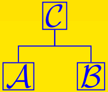
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.
Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.



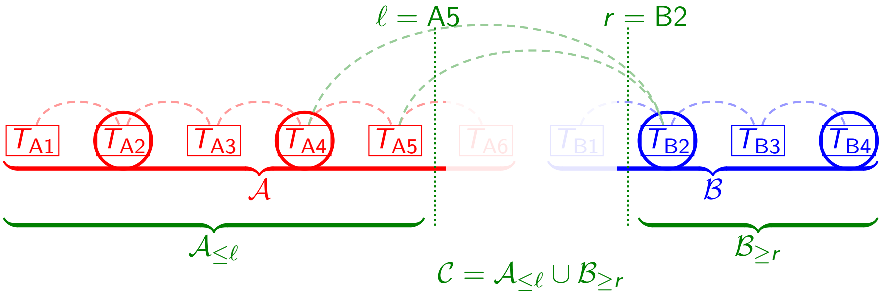
Merging consistent candidate bags



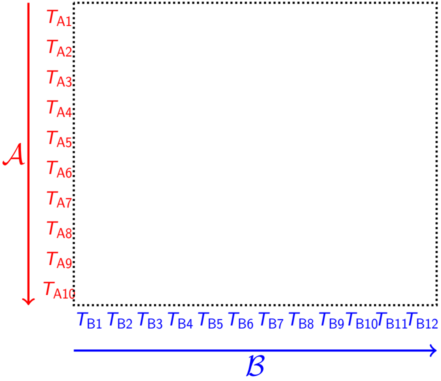
Input: Two consistent candidate sets $\mathcal{A}, \mathcal{B}$ s.t. $\max \mathcal{A} < \min \mathcal{B}$.
Output: consistent set $\mathcal{C}$ s.t. $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq \mathcal{C} \subseteq \mathcal{A} \cup \mathcal{B}$
where $\widehat{\mathcal{A}}$ is the occurrence posistions in $\mathcal{A}$.

Lemma (Consistency is one-way transitive)

If (T_i, T_j) and (T_j, T_k) are respectively consistent, where $i < j < k$, then $\{T_i, T_j, T_k\}$ is consistent.

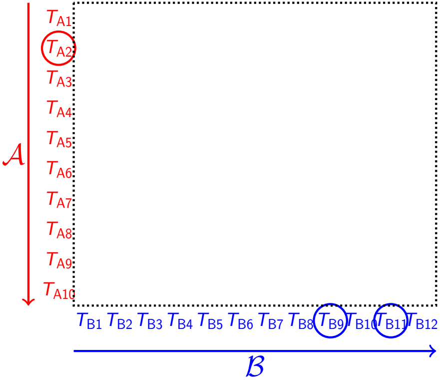


Duel grid



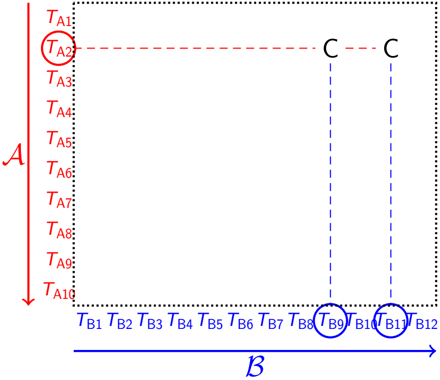
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



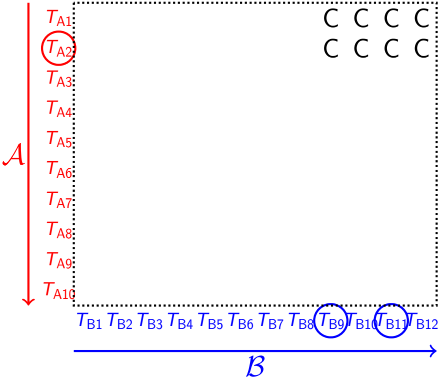
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



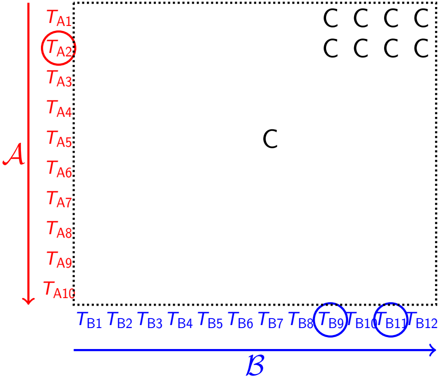
$$\mathfrak{D}(\mathsf{A}i, \mathsf{B}j) = \begin{cases} \mathsf{C} & \text{if } (T_{\mathsf{A}i}, T_{\mathsf{B}j}) \text{ is consistent,} \\ \mathsf{A} & \text{if } T_{\mathsf{A}i} \text{ wins against } T_{\mathsf{B}j}, \\ \mathsf{B} & \text{if } T_{\mathsf{B}j} \text{ wins against } T_{\mathsf{A}i}. \end{cases}$$

Duel grid



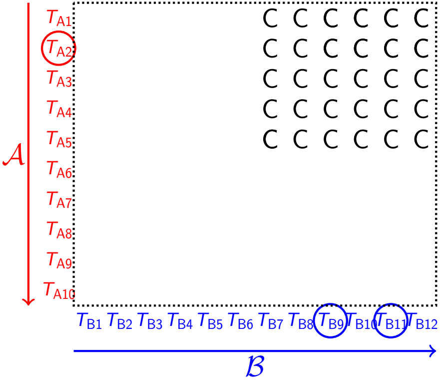
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



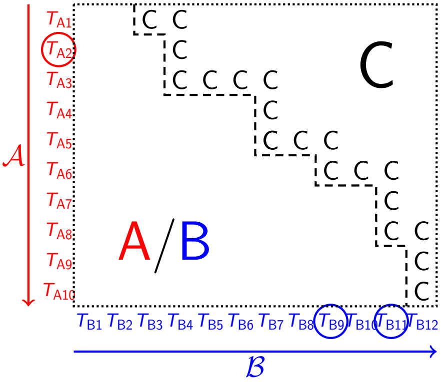
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



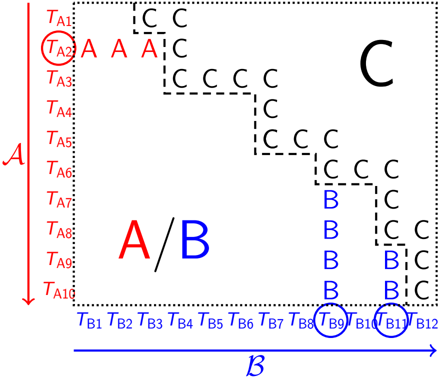
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



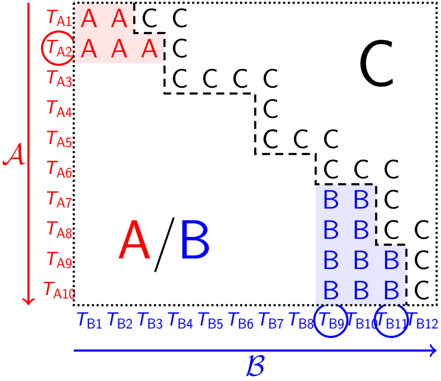
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



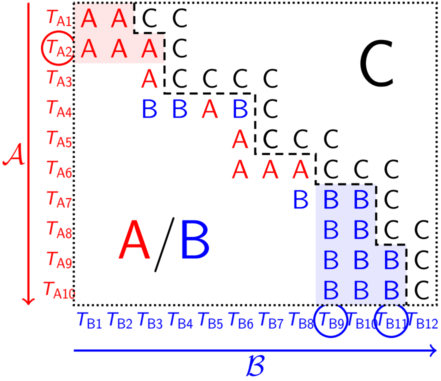
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



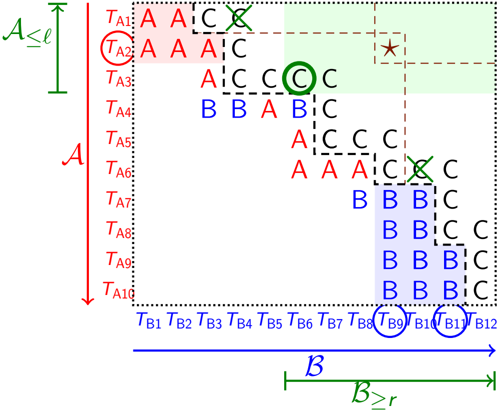
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid



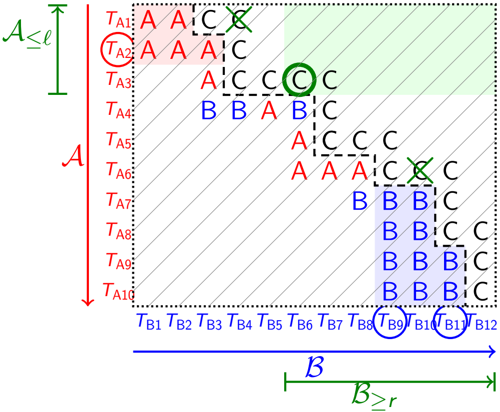
$$\mathfrak{D}(\mathrm{A}i, \mathrm{B}j) = \begin{cases} \mathrm{C} & \text{if } (T_{\mathrm{A}i}, T_{\mathrm{B}j}) \text{ is consistent,} \\ \mathrm{A} & \text{if } T_{\mathrm{A}i} \text{ wins against } T_{\mathrm{B}j}, \\ \mathrm{B} & \text{if } T_{\mathrm{B}j} \text{ wins against } T_{\mathrm{A}i}. \end{cases}$$

Duel grid

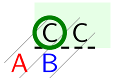


Output: $\mathcal{C} = \mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\geq r}$

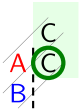
Duel grid



Find AC-BC transit point (ℓ, r) on the borderline by two-fold binary search.



or

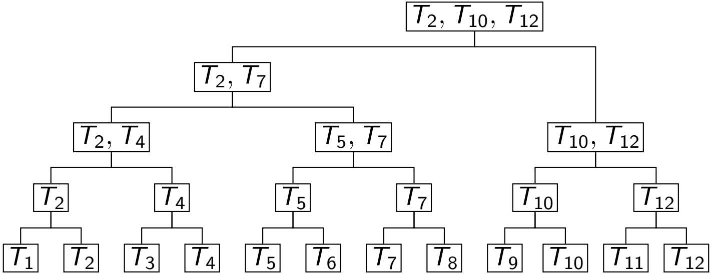


We never miss occurrences

Output: $\mathcal{C} = \mathcal{A}_{\le \ell} \cup \mathcal{B}_{\ge r}$

Dueling stage summary

- All occurrences survive
- Survivors are consistent



Lemma (Dueling stage)

The dueling stage runs in $\mathrm{O}(\xi_n^{\mathrm{t}} \log n \log^2 m)$ *time and* $\mathrm{O}(\xi_n^{\mathrm{w}} n \log^2 m)$ *work on P-CRCW-PRAM.*

Result

- There may be many survivors, but they are pairwise consistent.
- Overlapping parts of candidates do not have to be scanned independently for each candidate.

Lemma (Sweeping stage)

The sweeping stage can be done in $O(\xi_n^{\mathrm{t}} \cdot \log n)$ *time and with* $O(\xi_n^{\mathrm{w}} \cdot n \log n)$ *work on the P-CRCW PRAM.*

Result

- There may be many survivors, but they are pairwise consistent.
- Overlapping parts of candidates do not have to be scanned independently for each candidate.

Lemma (Sweeping stage)

The sweeping stage can be done in $O(\xi_n^{\mathrm{t}} \cdot \log n)$ time and with $O(\xi_n^{\mathrm{w}} \cdot n \log n)$ work on the P-CRCW PRAM.

Theorem

Given P and T, all the $\approx$-matching positions can be found in $\mathrm{O}(\tau_m^{\mathrm{t}} + \xi_m^{\mathrm{t}} \cdot \log^3 m)$ time and with $\mathrm{O}(\tau_m^{\mathrm{w}} \cdot n/m + \xi_m^{\mathrm{w}} \cdot n \log^2 m)$ work on the P-CRCW PRAM.

(by matching between P and $T[i : i + 2m - 1]$ for $i = 1, m + 1, 2m + 1, \dots,$ in parallel)

Discussions

Our algorithm relies on technical properties on SCER-encoding.

- $|X| = \phi(|X|)$
- $X \approx Y$ iff $\phi(X) = \phi(Y)$
- $\phi(X)[1:i] = \phi(X[1:i])$
- $\phi(X)[i] = \phi(Y)[i]$ implies $\phi(X[j+1:i])[i-j] = \phi(Y[j+1:i])[i-j]$

$\sim$ If two positions show no mismatch, then they show no mismatch by removing prefixes.

Discussions

Our algorithm relies on technical properties on SCER-encoding.

- $|X| = \phi(|X|)$
- $X \approx Y$ iff $\phi(X) = \phi(Y)$
- $\phi(X)[1:i] = \phi(X[1:i])$
- $\phi(X)[i] = \phi(Y)[i]$ implies $\phi(X[j+1:i])[i-j] = \phi(Y[j+1:i])[i-j]$

$\sim$ If two positions show no mismatch, then they show no mismatch by removing prefixes.

- The standard encoding pred (nearest neighbor encoding) for order-isomorphic matching does not satisfy the last condition.
  - For $X = 1\ 3\ 2$, $Y = 2\ 1\ 3$, $\mathrm{pred}(X)[3] = \mathrm{pred}(Y)[3]$, but $\mathrm{pred}(5\ 3)[2] \neq \mathrm{pred}(1\ 3)[2]$.
- Parallel OI matching based on duel-&-sweep with pred is possible (Jargalsaikhan 2022, PhD Thesis)
- Better generalization?