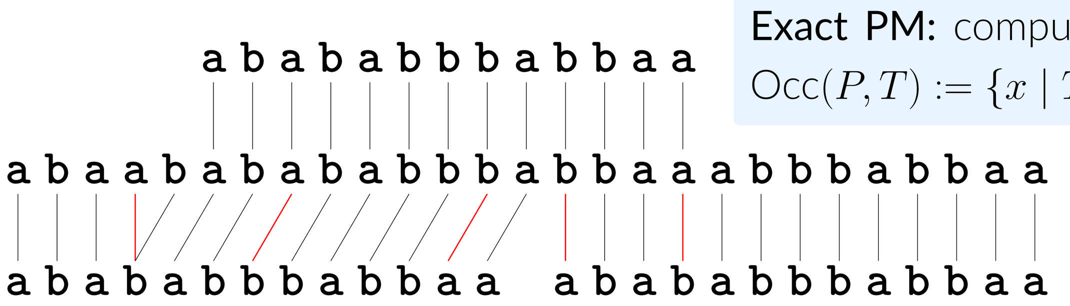
On the Communcation Complexity of Approximate Pattern Matching

Tomasz Kociumaka[1], **Jakob Nogler**[2], and Philip Wellnitz[3]

[1]Max Planck Institute for Informatics, SIC, Germany [2]ETH Zurich, Switzerland [3]National Institute of Informatics, SOKENDAI, Japan

Pattern Matching (PM)

Given a *text* T and a *pattern* P of length $|T| = n$ and $|P| = m$, find all substrings of T where P occurs exactly (exact PM) or with few errors (approximate PM).



Exact PM: compute $\text{Occ}(P,T) := \{x \mid T[x\,..\,x+m) = P\}$.

Interesting case: $n \le 3/2 \cdot m$.

PM with edits: compute $\text{Occ}^E_k(P,T) := \{x \mid \exists y\ \text{ED}(T[x\,..\,y), P) \le k\}$.

PM with mismatches: compute $\text{Occ}^H_k(P,T) := \{x \mid \text{HD}(T[x\,..\,x+m), P) \le k\}$.

Communication Complexity

Alice → Bob: One-way protocol

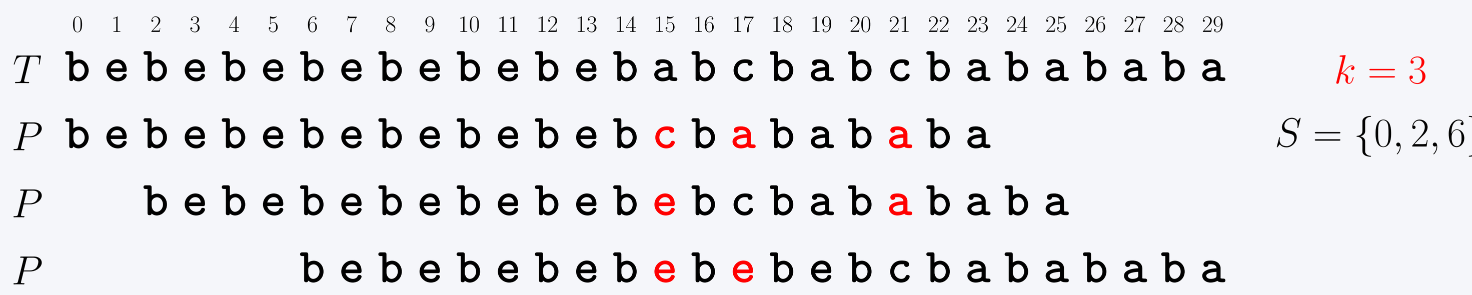
1. Alice receives a PM instance. Text T, Pattern P, Threshold k
2. Alice compresses the input.
3. Alice sends compressed data to Bob.
4. Bob needs to reconstructs the output of the instance. Set $\text{Occ}^E_k(P,T)$

Communication Complexity = "minimum # of machine words to send to Bob"

	UB	LB	Refererence
Exact PM	$\mathcal{O}(1)$	$\Omega(1)$	Periodicity Lemma, [FW65]
PM with mismatches	$\mathcal{O}(k)$	$\Omega(k)$	[CKP19]
PM with edits	$\mathcal{O}(k^3)$		[CKW20]
PM with edits	$\boldsymbol{\mathcal{O}(k \log m)}$	$\boldsymbol{\Omega(k)}$	This Work

How to Achieve $\mathcal{O}(k \log m)$ for PM with Mismatches

How does Alice encode $\text{Occ}^H_k(P,T)$?



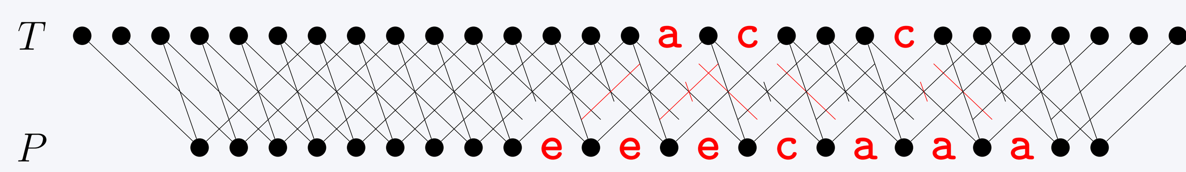
Alice sends S, $\{(15, \mathtt{c}, \mathtt{a}), (17, \mathtt{a}, \mathtt{c}), (21, \mathtt{a}, \mathtt{c})\}$, $\{(13, \mathtt{e}, \mathtt{a}), (19, \mathtt{a}, \mathtt{c})\}$, and $\{(9, \mathtt{e}, \mathtt{a}), (11, \mathtt{e}, \mathtt{c})\}$.

1. Crops T s.t. $\{0, n-m\} \subseteq \text{Occ}^H_k(P,T)$.
2. Selects set $\{0, n-m\} \subseteq S \subseteq \text{Occ}^H_k(P,T)$ s.t. $\gcd(S) = \gcd(\text{Occ}^H_k(P,T)) =: g$.
3. Sends S and the *mismatch information* for every $i \in S$ to Bob, i.e.,
$$\{(j, P[j], T[i+j]) \mid P[j] \ne T[i+j]\}.$$

Claim: Alice can choose S s.t. $|S| = \mathcal{O}(\log m)$.

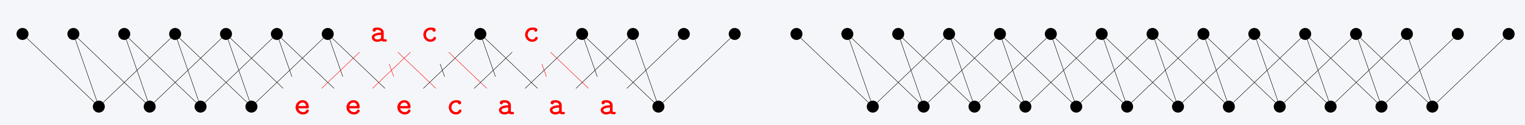
How does Bob decode Alice's message?

1. Bob constructs the graph $\mathbf{G}_S$. V = characters of P & T, $E = \{\{P[j], T[i+j]\} \mid i \in S, j \in [0\,..\,m)\}$



Red edges correspond to mismatches.

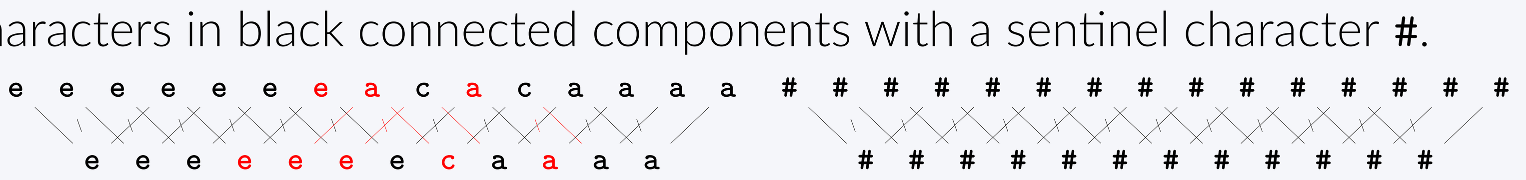
2. Bob divides $\mathbf{G}_S$ into *black and red connected components*.



Red component (at least one red edge). Back component (no red edges).

Claim: For every remainder $c \in [0\,..\,g)$ modulo g there is a connected components consisting of all vertices $P[i]$ and $T[i]$ with $i \equiv_g c$.

3. Bob propagates characters in red connected components, and replaces characters in black connected components with a sentinel character #.



4. Bob thereby obtains $T^\#$ and $P^\#$.

$T^\#$ # e # e # e # e # e # e # e # e # a # c # a # c # a # a # a # a

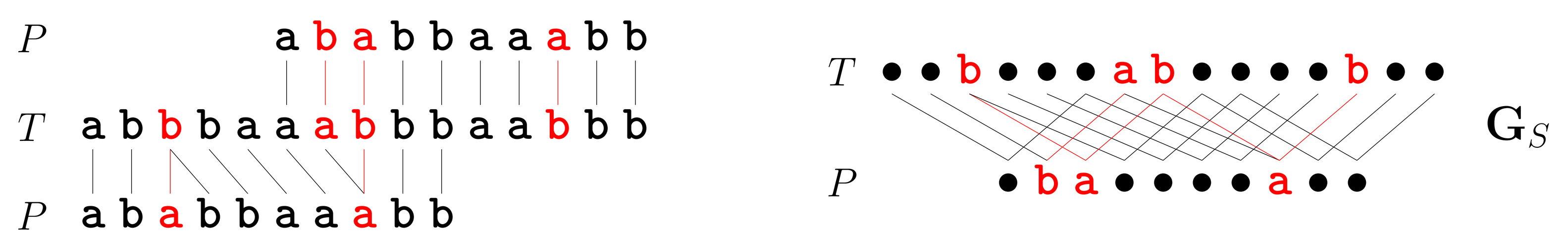
$P^\#$ # e # e # e # e # e # e # e # c # a # a # a # a

5. Bob returns $\text{Occ}^H_k(P^\#, T^\#)$.

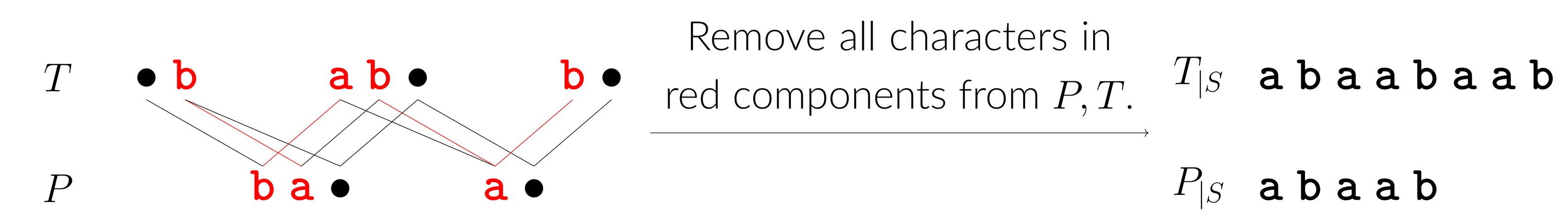
Claim: $\text{Occ}^H_k(P^\#, T^\#) = \text{Occ}^H_k(P,T)$.

Extending the Techinques to PM with Edits

We store a set S of $\mathcal{O}(\log m)$ k-edits occurences (including a prefix and a suffix), along with the information for their edits. We use them to construct a graph $\mathbf{G}_S$.



What changes?



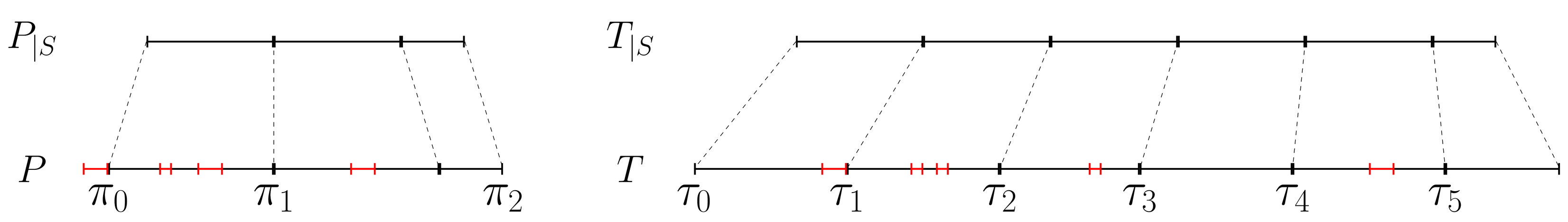
Red connected components do not have a periodic structure anymore.

Black connected components have a periodic structure in $P_{|S}$ and $T_{|S}$.

Claim: $P_{|S}$, $T_{|S}$ are periodic with period $\text{bc}(\mathbf{G}_S)$ (# number of black components). Moreover, for every remainder $c \in [0\,..\,\text{bc}(\mathbf{G}_S))$, there is a connected components consisting of all vertices $P[i]$ and $T[i]$ with $i \equiv_{\text{bc}(\mathbf{G}_S)} c$.

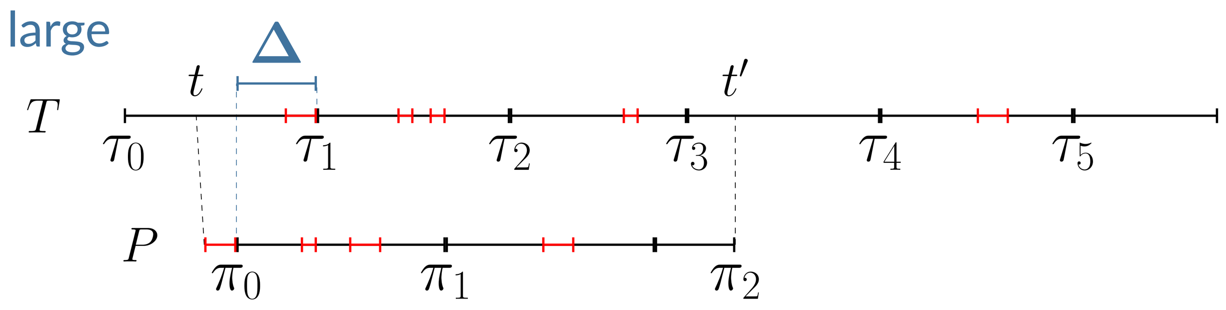
How do we keep S small? We add alignments iteratively to S. For each added alignment we aim to at least halve the period of $P_{|S}$ and $T_{|S}$.

For which new alignments added to S does the period of $P_{|S}$ and $T_{|S}$ decrease?



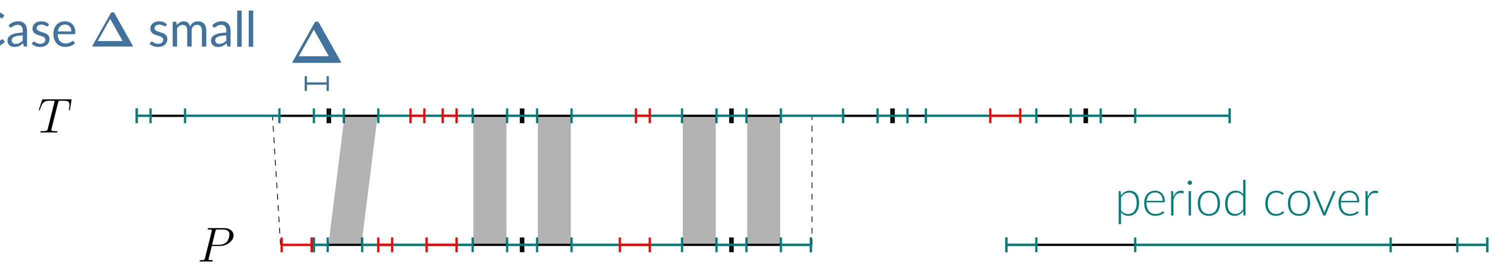
For a new k-edit alignment $P \rightsquigarrow T[t\,..\,t')$ we consider $\Delta = \min_i |\tau_i - t - \pi_0|$.

(1) Case Δ large



Then, any node in a black components is far away from any another node in the same component. If the alignment is added to S, every black component (if it not becomes red) is merged with another one

(2) Case Δ small



Then, we learn a set of character contained in black components around characters in red components (*period cover*). Characters outside from the period cover always match with characters belonging to the same black component.

Why does this work? New results linking "close alignments" and compressibility (see for example [CKW23], [GJKT24], [GK24]).

Application: PM with Edits in the Quantum Setting

There is a quantum algorithm that, given a PM with edits instance:

- computes $\text{Occ}^E_k(P,T)$ using $\widehat{\mathcal{O}}(n/m \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(n/m \cdot (\sqrt{km} + k^{3.5}))$ time;
- decides whether $\text{Occ}^E_k(P,T) \ne \emptyset$ using $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot (\sqrt{km} + k^{3.5}))$ time.

The number of queries is optimal for small k (up to a subpolynomial factor).

Main ingredients of the algorithm:

- Structural insights for approximate pattern matching [CKW20].
- Quantum algorithm for computing bounded edit distance [GJKT24].
- Quantum *Gap Edit Distance* algorithm (adapted from [GKKS22]).

Where do we use the previous results?
Suppose that we have a set $\text{Occ}^E_k(P,T) \subseteq H \subseteq [0\,..\,n)$ of *candidate positions*. We can retrieve $\text{Occ}^E_k(P,T)$ by only testing $\mathcal{O}(\log n)$ of the positions in H:

1. Keep a set of K-edit alignments for $K > k$ (but not too large).
2. Discard positions in H "covered" by S. For the others run a (k, K)-Gap Edit Distance oracle (faster than computing the edit distance).
3. If there is one with dist. $\le K$, retrieve the edits and repeat (2) until $H = \emptyset$.