Optimal Algorithms for Bounded Weighted Edit Distance

Tomasz Kociumaka

max planck institut informatik

SIC Saarland Informatics Campus

Based on joint works with:

Debarati Das, Jacob Gilbert, MohammadTaghi Hajiaghayi, and Barna Saha

PennState · University of Maryland · UC San Diego

Alejandro Cassis and Philip Wellnitz

Universität des Saarlandes · max planck institut informatik · SIC Saarland Informatics Campus

ADFOCS 2023, August 23rd, 2023

(Weighted) Edit Distance

Edit distance ed(X, Y) — Levenshtein distance

Minimum number of character insertions, deletions, and substitutions that transform X to Y.

X : `b b a b a b b a a b`

Y : `b a b a b a a a b b`

$\mathrm{ed}(X, Y) = 3$

(Weighted) Edit Distance

Edit distance $\mathrm{ed}(X, Y)$ — Levenshtein distance

Minimum number of character insertions, deletions, and substitutions that transform X to Y.

w	ε	a	b
ε	0	1	1
a	1	0	1
b	1	1	0

X : b b a b a b b a a b

Y : b a b a b a a a b b

$\mathrm{ed}^w(X, Y) = 3$

Weighted edit distance $\mathrm{ed}^w(X, Y)$ — $w : (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to \mathbb{R}_{\geq 0}$

The minimum cost of transforming X to Y using character edits, where:

- inserting b costs $w(\varepsilon, b)$;
- deleting a costs $w(a, \varepsilon)$;
- substituting a for b costs $w(a, b)$.

(Weighted) Edit Distance

Edit distance ed(X, Y) — Levenshtein distance

Minimum number of character insertions, deletions, and substitutions that transform X to Y.

w	ε	a	b
ε	0	1	3
a	1	0	2
b	3	2	0

X : b b a b a b b a a b

Y : b a b a b a a a b b

$\mathrm{ed}^w(X, Y) \leq 8$

Weighted edit distance $\mathrm{ed}^w(X, Y)$ — $w : (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to \mathbb{R}_{\geq 0}$

The minimum cost of transforming X to Y using character edits, where:

- inserting b costs $w(\varepsilon, b)$;
- deleting a costs $w(a, \varepsilon)$;
- substituting a for b costs $w(a, b)$.

(Weighted) Edit Distance

Edit distance $\mathrm{ed}(X, Y)$ — Levenshtein distance

Minimum number of character insertions, deletions, and substitutions that transform X to Y.

w	ε	a	b
ε	0	1	3
a	1	0	2
b	3	2	0

X : b b a b a b b a a b

Y : b a b a b a a a b b

$\mathrm{ed}^w(X, Y) = 6$

Weighted edit distance $\mathrm{ed}^w(X, Y)$ — $w : (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to \mathbb{R}_{\geq 0}$

The minimum cost of transforming X to Y using character edits, where:

- inserting b costs $w(\varepsilon, b)$;
- deleting a costs $w(a, \varepsilon)$;
- substituting a for b costs $w(a, b)$.

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Bounded edit distance: $\mathrm{ed}^w(X, Y) \leq k$ for some input threshold k

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Bounded edit distance: $\mathrm{ed}^w(X, Y) \le k$ for some input threshold k

Assumption: Normalized weight function, that is, $w(a, b) \ge 1$ for all distinct $a, b \in \Sigma \cup \{\varepsilon\}$.

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Bounded edit distance: $\mathrm{ed}^w(X, Y) \le k$ for some input threshold k

Assumption: Normalized weight function, that is, $w(a, b) \ge 1$ for all distinct $a, b \in \Sigma \cup \{\varepsilon\}$.

Ukk85,Mye86	$\mathcal{O}(nk)$	
LV88	$\mathcal{O}(n + k^2)$	unweighted only
folklore	$\Omega(n + k^{2-o(1)})$	unweighted; conditioned on SETH/OV

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Bounded edit distance: $\text{ed}^w(X, Y) \leq k$ for some input threshold k

Assumption: Normalized weight function, that is, $w(a, b) \geq 1$ for all distinct $a, b \in \Sigma \cup \{\varepsilon\}$.

Reference	Time	Remarks
Ukk85,Mye86	$\mathcal{O}(nk)$	
LV88	$\mathcal{O}(n + k^2)$	unweighted only
folklore	$\Omega(n + k^{2-o(1)})$	unweighted; conditioned on SETH/OV
DGH**K**S23	$\mathcal{O}(n + k^5)$	

Computing Edit Distance

Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

Bounded edit distance: $\mathrm{ed}^w(X, Y) \le k$ for some input threshold k

Assumption: Normalized weight function, that is, $w(a, b) \ge 1$ for all distinct $a, b \in \Sigma \cup \{\varepsilon\}$.

Ukk85,Mye86	$\mathcal{O}(nk)$	
LV88	$\mathcal{O}(n + k^2)$	unweighted only
folklore	$\Omega(n + k^{2-o(1)})$	unweighted; conditioned on SETH/OV
DGH**K**S23	$\mathcal{O}(n + k^5)$	
C**K**W23	$\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$	

Computing Edit Distance

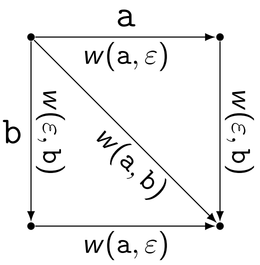
Reference	Time	Remarks
Vin68,NW70,Sel74,WF74	$\mathcal{O}(n^2)$	
BI18	$\Omega(n^{2-o(1)})$	unweighted; conditioned on SETH/OV

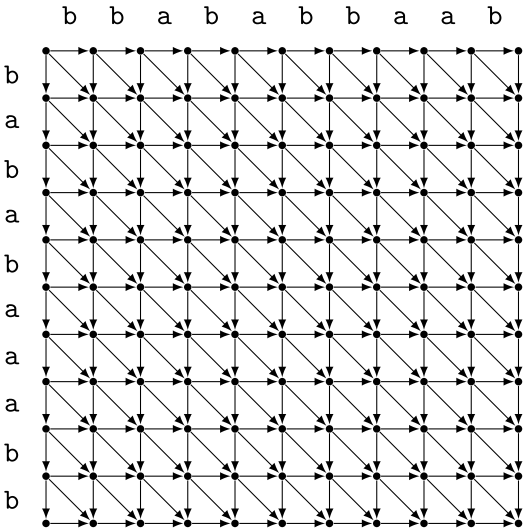
Bounded edit distance: $\mathrm{ed}^w(X, Y) \le k$ for some input threshold k

Assumption: Normalized weight function, that is, $w(a, b) \ge 1$ for all distinct $a, b \in \Sigma \cup \{\varepsilon\}$.

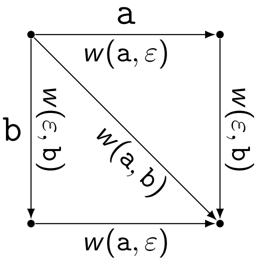
Reference	Time	Remarks
Ukk85,Mye86	$\mathcal{O}(nk)$	
LV88	$\mathcal{O}(n + k^2)$	unweighted only
folklore	$\Omega(n + k^{2-o(1)})$	unweighted; conditioned on SETH/OV
DGH**K**S23	$\mathcal{O}(n + k^5)$	
C**K**W23	$\tilde{\mathcal{O}}(n + \sqrt{nk^3})$	
C**K**W23	$\Omega(n + \sqrt{nk^{3-o(1)}})$	conditioned on APSP, $\sqrt{n} \le k \le n$

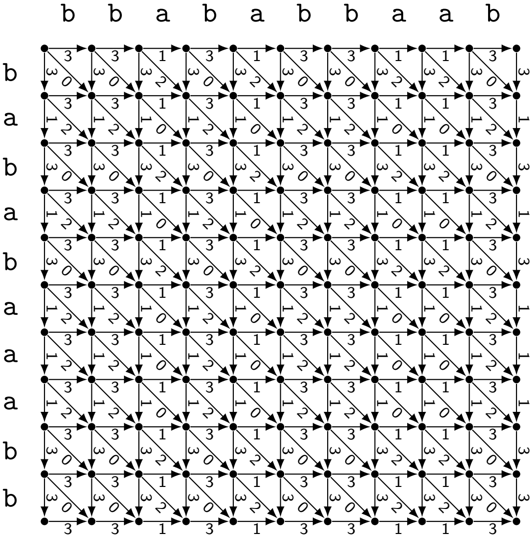
Alignment Graph and Dynamic-Programming Algorithms



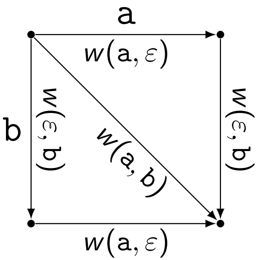


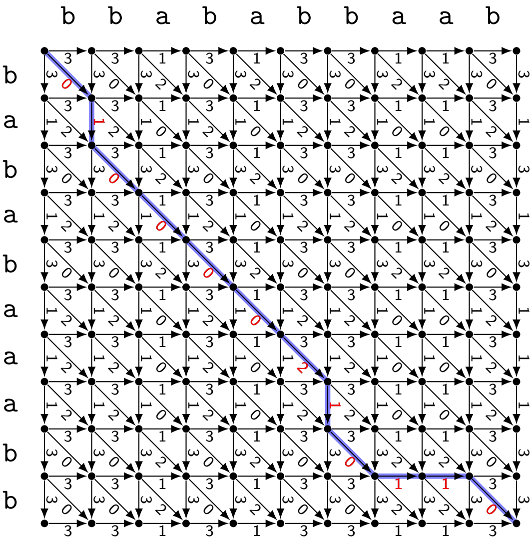
Alignment Graph and Dynamic-Programming Algorithms

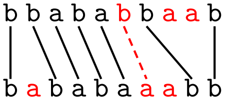




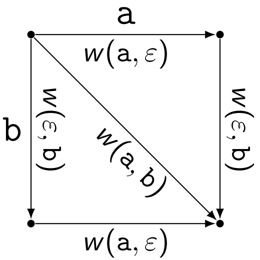
Alignment Graph and Dynamic-Programming Algorithms



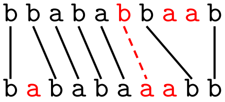




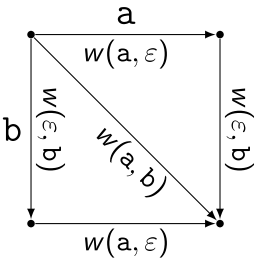
Alignment Graph and Dynamic-Programming Algorithms



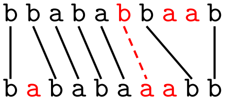
		b	b	a	b	a	b	b	a	a	b
	0	3	6	7	10	11	14	17	18	19	22
b	3	0	3	4	7	8	11	14	15	16	19
a	4	1	2	3	6	7	10	13	14	15	18
b	7	4	1	2	3	4	7	10	11	12	15
a	8	5	2	1	4	3	6	9	10	11	14
b	11	8	5	4	1	2	3	6	7	8	11
a	12	9	6	5	2	1	4	5	6	7	10
a	13	10	7	6	3	2	3	6	5	6	9
a	14	11	8	7	4	3	4	5	6	5	8
b	17	14	11	10	7	6	3	4	5	6	5
b	20	17	14	13	10	9	6	3	4	5	6



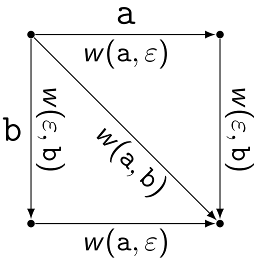
Alignment Graph and Dynamic-Programming Algorithms



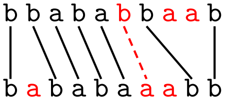
		b	b	a	b	a	b	b	a	a	b
	0	3	6	7	10	11	14	17	18	19	22
b	3	0	3	4	7	8	11	14	15	16	19
a	4	1	2	3	6	7	10	13	14	15	18
b	7	4	1	2	3	4	7	10	11	12	15
a	8	5	2	1	4	3	6	9	10	11	14
b	11	8	5	4	1	2	3	6	7	8	11
a	12	9	6	5	2	1	4	5	6	7	10
a	13	10	7	6	3	2	3	6	5	6	9
a	14	11	8	7	4	3	4	5	6	5	8
b	17	14	11	10	7	6	3	4	5	6	5
b	20	17	14	13	10	9	6	3	4	5	6



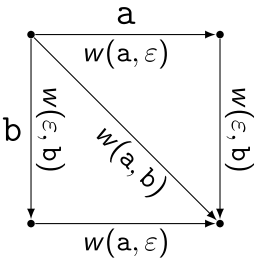
Alignment Graph and Dynamic-Programming Algorithms

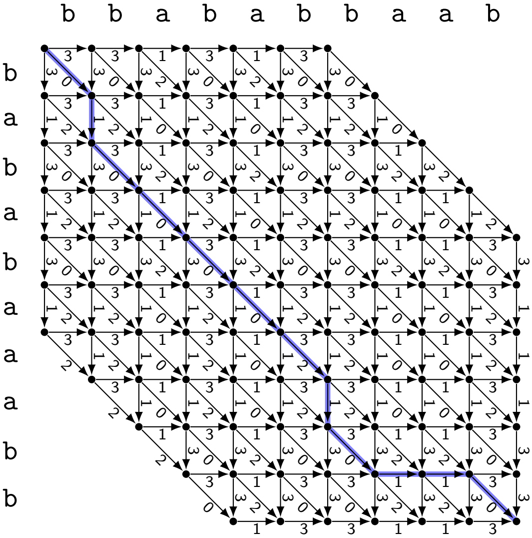


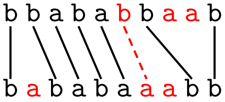
		b	b	a	b	a	b	b	a	a	b
	0	3	6	7	10	11	14				
b	3	0	3	4	7	8	11	14			
a	4	1	2	3	6	7	10	13	14		
b	7	4	1	2	3	4	7	10	11	12	
a	8	5	2	1	4	3	6	9	10	11	14
b	11	8	5	4	1	2	3	6	7	8	11
a	12	9	6	5	2	1	4	5	6	7	10
a		10	7	6	3	2	3	6	5	6	9
a			8	7	4	3	4	5	6	5	8
b				10	7	6	3	4	5	6	5
b					10	9	6	3	4	5	6



Alignment Graph and Dynamic-Programming Algorithms







Why Is Weigheted Edit Distance Harder?

Unweighted case:
The values along diagonals may only increase.

		b	b	a	b	a	b	b	a	a	b
	0	1	2	3	4	5	6	7	8	9	10
b	1	0	1	2	3	4	5	6	7	8	9
a	2	1	1	1	2	3	4	5	6	7	8
b	3	2	1	2	1	2	3	4	5	6	7
a	4	3	2	1	2	1	2	3	4	5	6
b	5	4	3	2	1	2	1	2	3	4	5
a	6	5	4	3	2	1	2	2	2	3	4
a	7	6	5	4	3	2	2	3	2	2	3
a	8	7	6	5	4	3	3	3	3	2	3
b	9	8	7	6	5	4	3	3	4	3	2
b	10	9	8	7	6	5	4	3	4	4	3

Why Is Weigheted Edit Distance Harder?

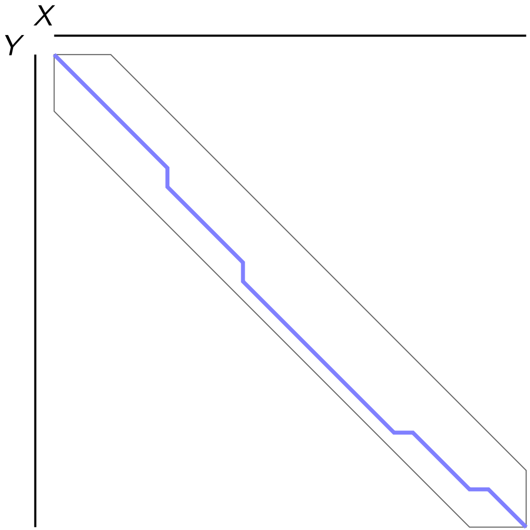
Unweighted case:
The values along diagonals may only increase.

Weighted case:
The values along diagonals may both increase and decrease.

		b	b	a	b	a	b	b	a	a	b
	0	3	6	7	10	11	14	17	18	19	22
b	3	0	3	4	7	8	11	14	15	16	19
a	4	1	2	3	6	7	10	13	14	15	18
b	7	4	1	2	3	4	7	10	11	12	15
a	8	5	2	1	4	3	6	9	10	11	14
b	11	8	5	4	1	2	3	6	7	8	11
a	12	9	6	5	2	1	4	5	6	7	10
a	13	10	7	6	3	2	3	6	5	6	9
a	14	11	8	7	4	3	4	5	6	5	8
b	17	14	11	10	7	6	3	4	5	6	5
b	20	17	14	13	10	9	6	3	4	5	6

Synchronized Fragments

Normalization implies $\mathrm{ed}(X, Y) \leq \mathrm{ed}^w(X, Y)$, so we build an optimal unweighted alignment.



Synchronized Fragments

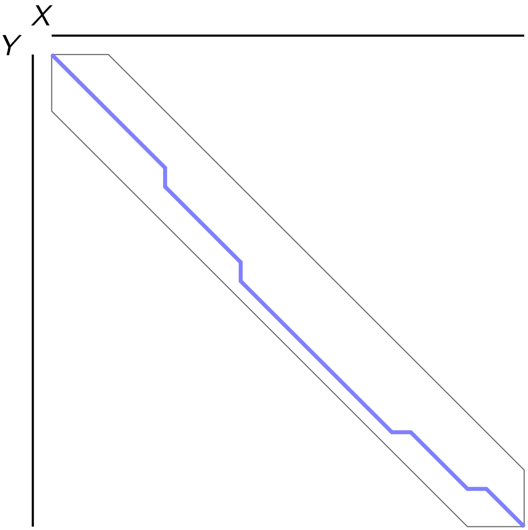
Normalization implies $\mathrm{ed}(X, Y) \le \mathrm{ed}^w(X, Y)$, so we build an optimal unweighted alignment.

The alignment decomposes X and Y into $\mathcal{O}(k)$ characters and synchronized fragments:

Synchronized fragments:

Two fragments $X[x \mathinner{.\,.} x')$ and $Y[y \mathinner{.\,.} y')$ are **k-synchronized** if

1. $|x - y| \le k$ and
2. $X[x \mathinner{.\,.} x') = Y[y \mathinner{.\,.} y')$.



Synchronized Fragments

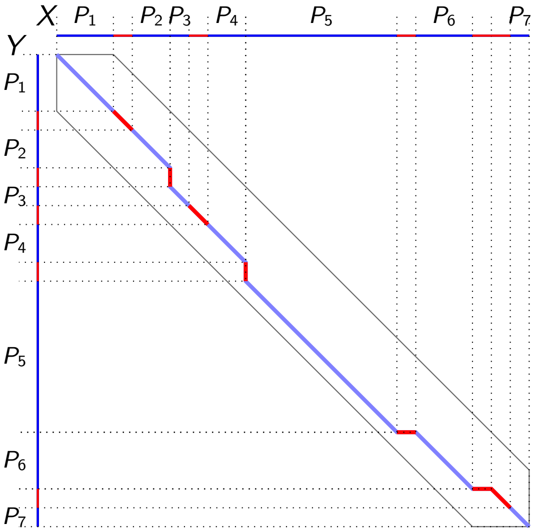
Normalization implies $\text{ed}(X, Y) \le \text{ed}^w(X, Y)$, so we build an optimal unweighted alignment.

The alignment decomposes X and Y into $\mathcal{O}(k)$ characters and synchronized fragments:

Synchronized fragments:

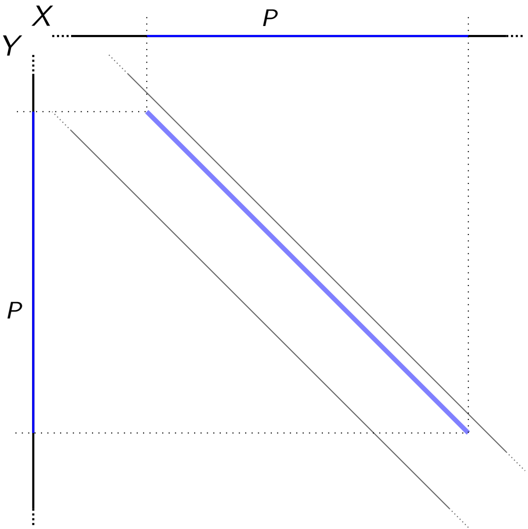
Two fragments $X[x\,..\,x')$ and $Y[y\,..\,y')$ are ***k*-synchronized** if

1. $|x - y| \le k$ and
2. $X[x\,..\,x') = Y[y\,..\,y')$.



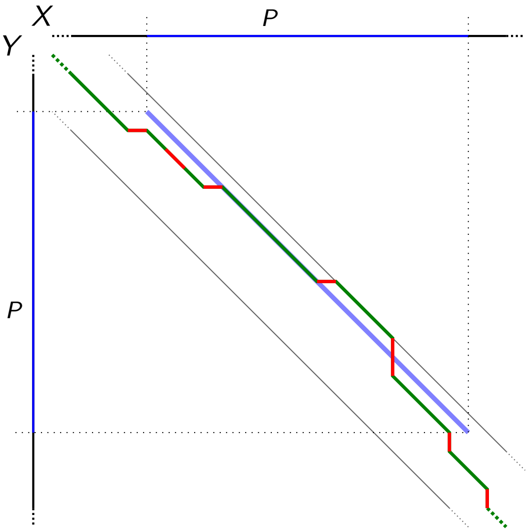
Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?



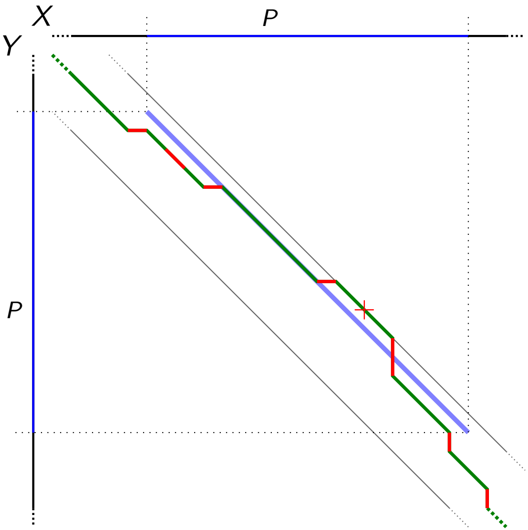
Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?



Synchronized Fragments vs Optimal Alignments

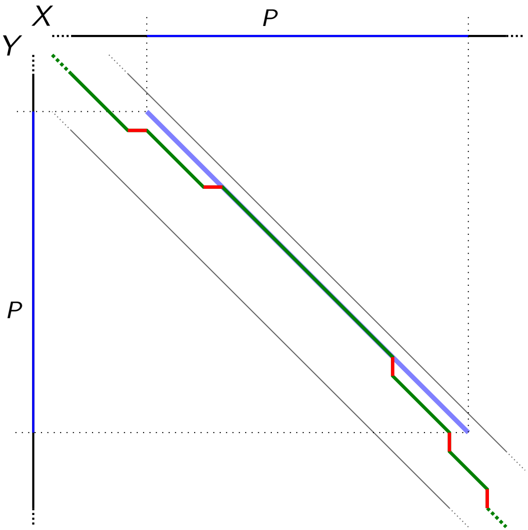
How can an optimal weighted alignment interact with synchronized fragments?



Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?

- The alignment touches the corresponding segment at most once.

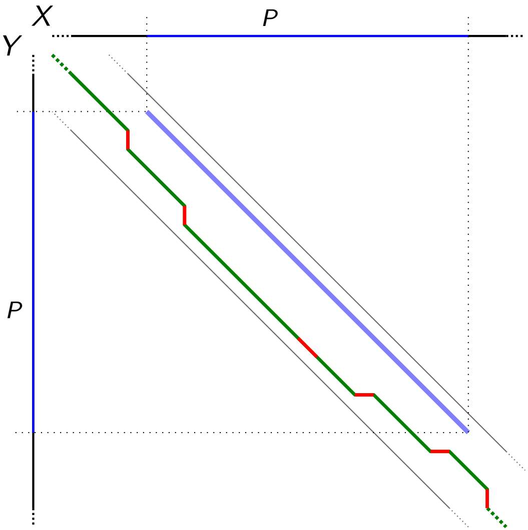


Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?

- The alignment touches the corresponding segment at most once.

What if the alignment never touches the segment?



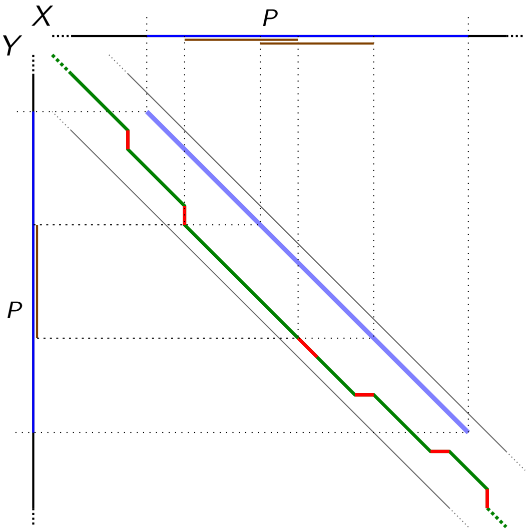
Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?

- The alignment touches the corresponding segment at most once.

What if the alignment never touches the segment?

- Each piece of Y matched perfectly occurs in X twice, $\leq 2k$ positions apart.



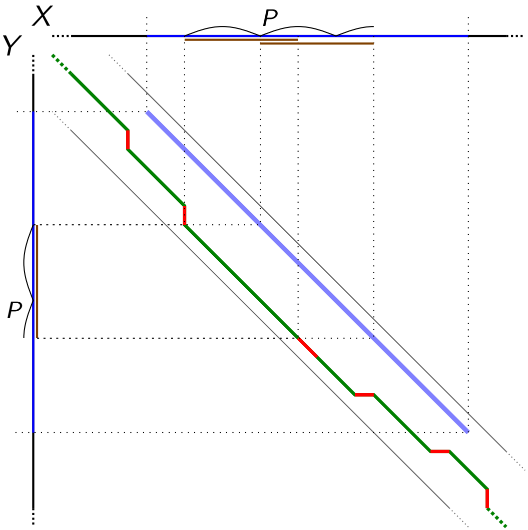
Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?

- The alignment touches the corresponding segment at most once.

What if the alignment never touches the segment?

- Each piece of Y matched perfectly occurs in X twice, $\leq 2k$ positions apart.



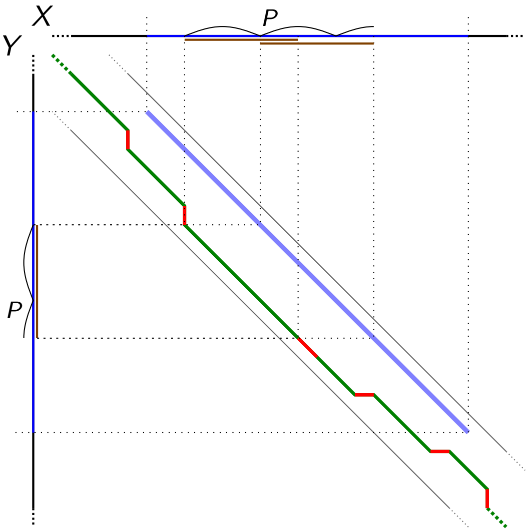
Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized fragments?

- The alignment touches the corresponding segment at most once.

What if the alignment never touches the segment?

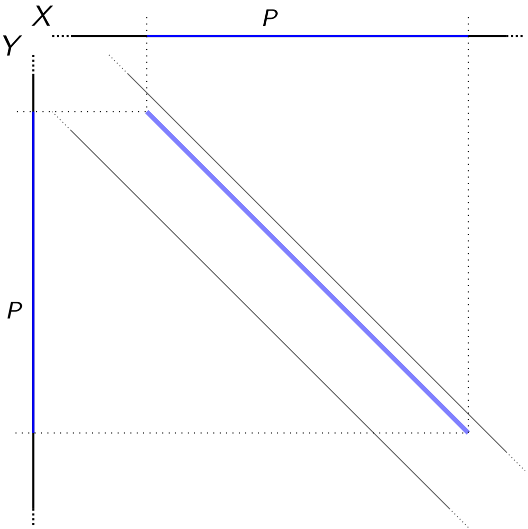
- Each piece of Y matched perfectly occurs in X twice, $\leq 2k$ positions apart.
- The synchronized fragments consist of $\leq 3k$ pieces with **periods** $\leq 2k$.



First Reduction

Lemma

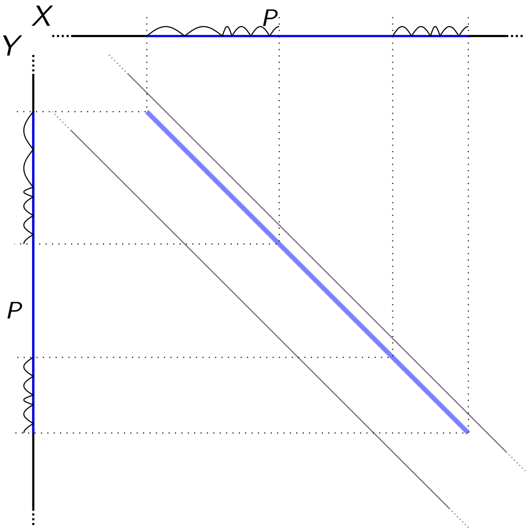
An *optimal weighted alignment* *matches* *synchronized fragments*, *except for a prefix and a suffix of* $\le 3k$ *pieces* *with periods* $\le 2k$.



First Reduction

Lemma

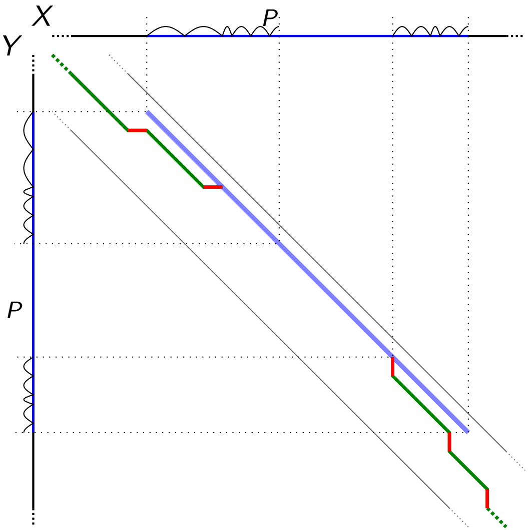
An optimal weighted alignment matches synchronized fragments, except for a prefix and a suffix of $\leq 3k$ *pieces with periods* $\leq 2k$.



First Reduction

Lemma

An optimal weighted alignment matches synchronized fragments, except for a prefix and a suffix of $\leq 3k$ *pieces with periods* $\leq 2k$.



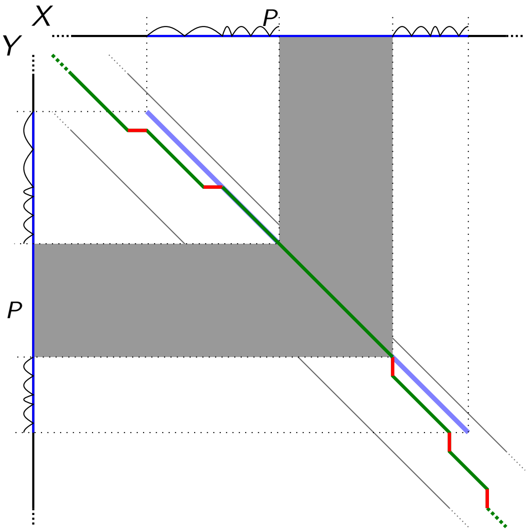
First Reduction

Lemma

An optimal weighted alignment matches synchronized fragments, except for a prefix and a suffix of $\leq 3k$ pieces with periods $\leq 2k$.

The characters in the middle, if any, can be removed without affecting

$$\mathrm{ed}^w_{\leq k}(X,Y) = \begin{cases} \mathrm{ed}^w(X,Y) & \text{if } \mathrm{ed}^w(X,Y) \leq k, \\ \infty & \text{otherwise.} \end{cases}$$



First Reduction

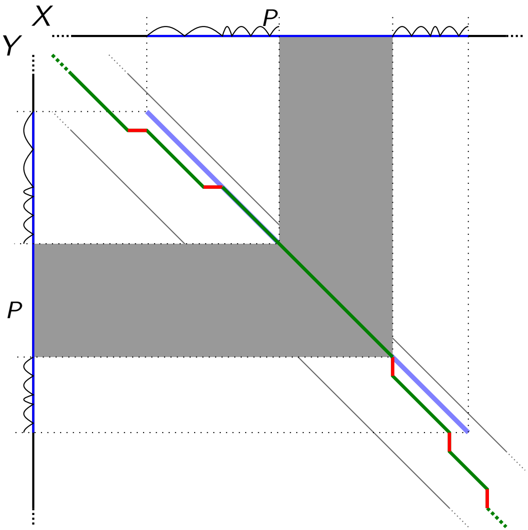
Lemma

An optimal weighted alignment matches synchronized fragments, except for a prefix and a suffix of $\leq 3k$ pieces with periods $\leq 2k$.

The characters in the middle, if any, can be removed without affecting

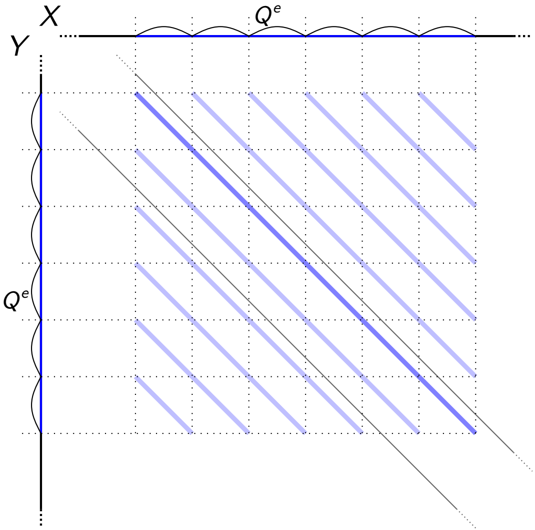
$$\mathrm{ed}^w_{\leq k}(X, Y) = \begin{cases} \mathrm{ed}^w(X, Y) & \text{if } \mathrm{ed}^w(X, Y) \leq k, \\ \infty & \text{otherwise.} \end{cases}$$

We can assume that X, Y can be decomposed into $\mathcal{O}(k^2)$ characters and synchronized fragments with periods $\leq 2k$.



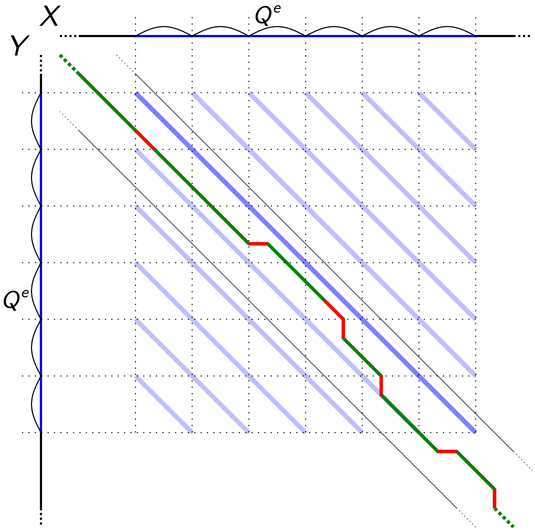
Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?



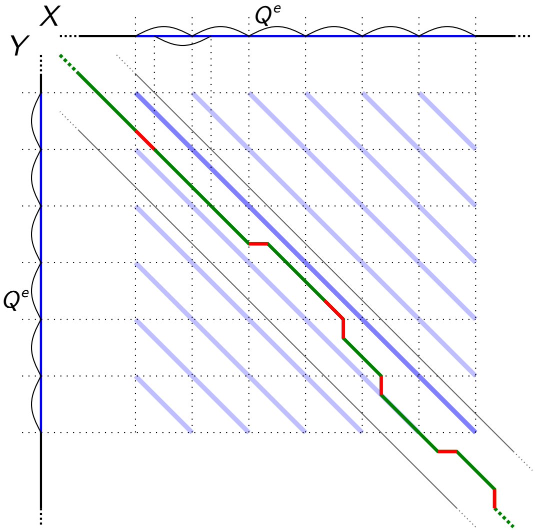
Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?



Periodic Synchronized Fragments vs Optimal Alignments

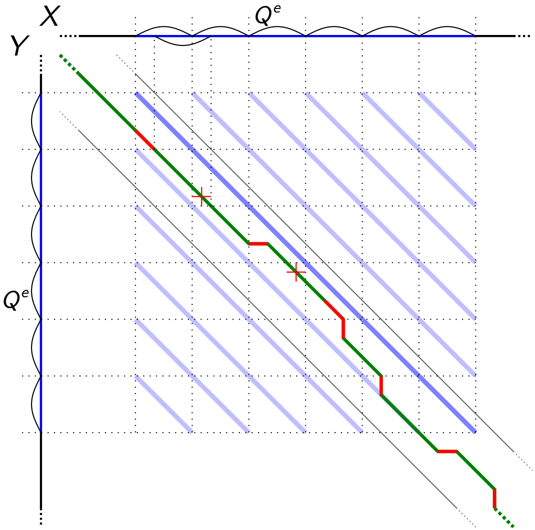
How can an optimal weighted alignment interact with synchronized occurrences of Q^e?



Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?

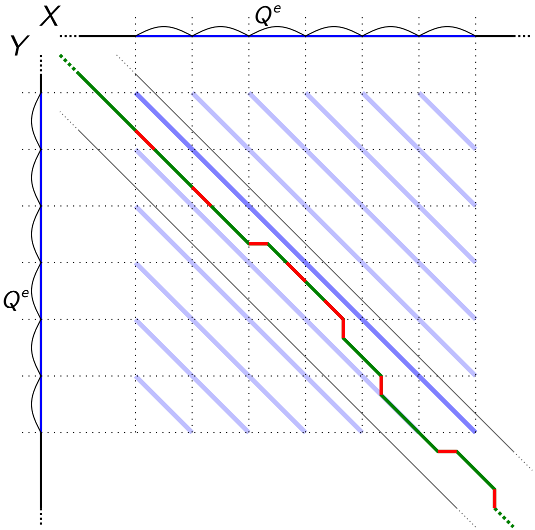
- If $|Q|$ consecutive characters are matched, they must be matched **canonically**.



Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?

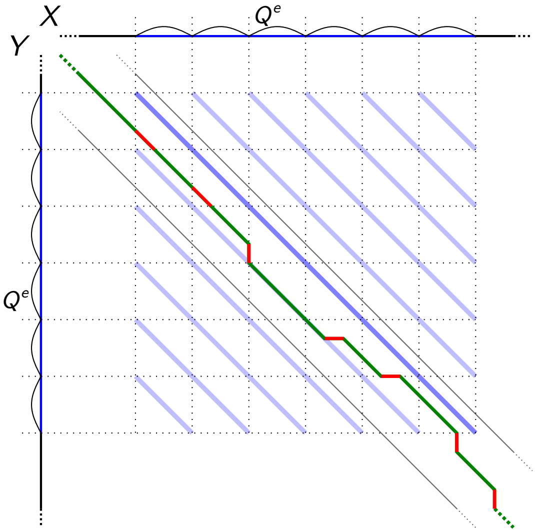
- If $|Q|$ consecutive characters are matched, they must be matched **canonically**.



Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?

- If $|Q|$ consecutive characters are matched, they must be matched **canonically**.
- If $e \geq 4k$, then at least $|Q|$ consecutive characters are matched canonically.

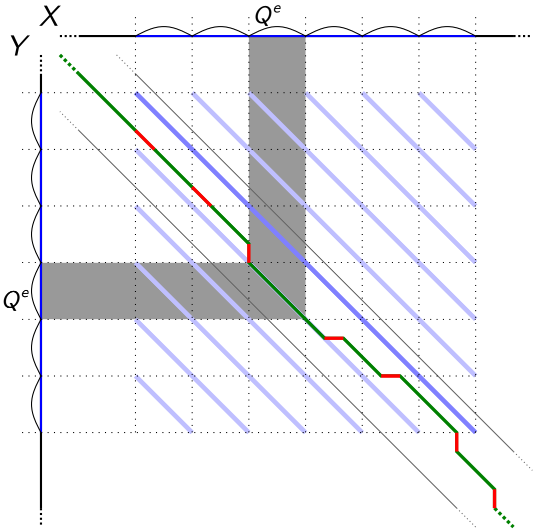


Periodic Synchronized Fragments vs Optimal Alignments

How can an optimal weighted alignment interact with synchronized occurrences of Q^e?

- If $|Q|$ consecutive characters are matched, they must be matched **canonically**.
- If $e \geq 4k$, then at least $|Q|$ consecutive characters are matched canonically.

If $e > 4k$, we can **reduce the exponent** to $4k$ without affecting $\mathrm{ed}^w_{\leq k}(X, Y)$.



Universal Kernel

Theorem (DGHK**S, STOC'23)**

There is an $\mathcal{O}(n)$ *time algorithm that, given strings* $X, Y \in \Sigma^{\leq n}$ *and an integer* $k > 0$*, constructs strings* X', Y' *of length* $\mathcal{O}(k^4)$ *such that* $\mathrm{ed}^w_{\leq k}(X', Y') = \mathrm{ed}^w_{\leq k}(X, Y)$ *holds for every normalized weight function* $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$.

Universal Kernel

Theorem (DGHKS, STOC'23)

There is an $\mathcal{O}(n)$ time algorithm that, given strings $X, Y \in \Sigma^{\le n}$ and an integer $k > 0$, constructs strings X', Y' of length $\mathcal{O}(k^4)$ such that $\mathrm{ed}^w_{\le k}(X', Y') = \mathrm{ed}^w_{\le k}(X, Y)$ holds for every normalized weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\ge 0}$.

1. Use the optimal unweighted alignment to decompose X and Y into $\mathcal{O}(k)$ characters and synchronized fragments.
2. For each pair of synchronized fragments:
   1. Find the longest prefix of $3k$ pieces with periods $\le 2k$.
   2. Find the longest suffix of $3k$ pieces with periods $\le 2k$.
   3. Remove the middle characters (between the prefix and the suffix), if any.
   4. For each periodic piece with exponent $e > 4k$, reduce the exponent to $4k$.

Universal Kernel

Theorem (DGHKS, STOC'23)

There is an $\mathcal{O}(n)$ time algorithm that, given strings $X, Y \in \Sigma^{\le n}$ and an integer $k > 0$, constructs strings X', Y' of length $\mathcal{O}(k^4)$ such that $\mathrm{ed}^w_{\le k}(X', Y') = \mathrm{ed}^w_{\le k}(X, Y)$ holds for every normalized weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\ge 0}$.

1. Use the optimal unweighted alignment to decompose X and Y into $\mathcal{O}(k)$ characters and synchronized fragments.
2. For each pair of synchronized fragments:
   a. Find the longest prefix of $3k$ pieces with periods $\le 2k$.
   b. Find the longest suffix of $3k$ pieces with periods $\le 2k$.
   c. Remove the middle characters (between the prefix and the suffix), if any.
   d. For each periodic piece with exponent $e > 4k$, reduce the exponent to $4k$.

$\mathcal{O}(k^4)$-size kernel $+$ $\mathcal{O}(nk)$-time dynamic-programming $\rightsquigarrow$ $\mathcal{O}(n + k^5)$-time algorithm

Optimal Algorithm for Bounded Weighted Edit Distance

Theorem (CK**W, FOCS'23)**

Given strings $X, Y \in \Sigma^{\leq n}$*, a threshold* $k \geq 0$*, and oracle access to a weight function* $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$*, the value* $\mathrm{ed}^w_{\leq k}(X, Y)$ *can be computed in* $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ *time.*

Optimal Algorithm for Bounded Weighted Edit Distance

Theorem (CK**W, FOCS'23)**

Given strings $X, Y \in \Sigma^{\leq n}$*, a threshold* $k \geq 0$*, and oracle access to a weight function* $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$*, the value* $\mathrm{ed}^w_{\leq k}(X, Y)$ *can be computed in* $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ *time.*

$\widetilde{\mathcal{O}}(n + k^4)$ **Multiple-Source Shortest Paths in Planar Graphs:**
Within a periodic piece, the alignment graph is repetitive.

Optimal Algorithm for Bounded Weighted Edit Distance

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, a threshold $k \geq 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, the value $\mathrm{ed}^w_{\leq k}(X, Y)$ can be computed in $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

$\widetilde{\mathcal{O}}(n + k^4)$ **Multiple-Source Shortest Paths in Planar Graphs:**
Within a periodic piece, the alignment graph is repetitive.

$\widetilde{\mathcal{O}}(n + k^3)$ **Divide and Conquer:**
Any single point of the optimal weighted alignment depends only on a context of $\mathcal{O}(k)$ pieces with period $\leq 2k$. Such a point splits the input into two halves.

Optimal Algorithm for Bounded Weighted Edit Distance

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\le n}$, a threshold $k \ge 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\ge 0}$, the value $\mathrm{ed}^w_{\le k}(X, Y)$ can be computed in $\tilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

$\tilde{\mathcal{O}}(n + k^4)$ **Multiple-Source Shortest Paths in Planar Graphs:**
Within a periodic piece, the alignment graph is repetitive.

$\tilde{\mathcal{O}}(n + k^3)$ **Divide and Conquer:**
Any single point of the optimal weighted alignment depends only on a context of $\mathcal{O}(k)$ pieces with period $\le 2k$. Such a point splits the input into two halves.

$\tilde{\mathcal{O}}(n + \sqrt{n^2 k^4})$ **Block periodicity $\to$ LZ compressibility:**
When two alignments are disjoint, the underlying fragments not only consists of $\mathcal{O}(k)$ periodic pieces, but also their Lempel–Ziv factorization has size $\mathcal{O}(k)$.

Optimal Algorithm for Bounded Weighted Edit Distance

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, a threshold $k \geq 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, the value $\mathrm{ed}^w_{\leq k}(X, Y)$ can be computed in $\tilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

$\tilde{\mathcal{O}}(n + k^4)$ **Multiple-Source Shortest Paths in Planar Graphs:**
Within a periodic piece, the alignment graph is repetitive.

$\tilde{\mathcal{O}}(n + k^3)$ **Divide and Conquer:**
Any single point of the optimal weighted alignment depends only on a context of $\mathcal{O}(k)$ pieces with period $\leq 2k$. Such a point splits the input into two halves.

$\tilde{\mathcal{O}}(n + \sqrt{n^2 k^4})$ **Block periodicity $\to$ LZ compressibility:**
When two alignments are disjoint, the underlying fragments not only consists of $\mathcal{O}(k)$ periodic pieces, but also their Lempel–Ziv factorization has size $\mathcal{O}(k)$.

$\tilde{\mathcal{O}}(n + \sqrt{nk^3})$ **LZ compressibility $\to$ self-edit distance:**
Replacing the Lempel–Ziv factorization with a tailor-made compressibility measure reveals even more repetitiveness of the alignment graph.

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, a threshold $k \geq 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, the value $\mathrm{ed}^{w}_{\leq k}(X, Y)$ can be computed in $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\le n}$*, a threshold* $k \ge 0$*, and oracle access to a weight function* $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\ge 0}$*, the value* $\mathrm{ed}^w_{\le k}(X, Y)$ *can be computed in* $\tilde{\mathcal{O}}(n + \sqrt{nk^3})$ *time.*

Theorem (CKW, FOCS'23)

Conditioned on the All-Pairs Shortest-Paths Hypothesis, the above running time is optimal (up to subpolynomial factors) for $\sqrt{n} \le k \le n$*.*

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, a threshold $k \geq 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, the value $\mathrm{ed}^w_{\leq k}(X, Y)$ can be computed in $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

Theorem (CKW, FOCS'23)

Conditioned on the All-Pairs Shortest-Paths Hypothesis, the above running time is optimal (up to subpolynomial factors) for $\sqrt{n} \leq k \leq n$.

Open Problems:

1. What is the true complexity of $\sqrt[3]{n} \leq k \leq \sqrt{n}$? Must be between $n + \sqrt{k^5}$ and $\sqrt{nk^3}$.

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, *a threshold* $k \geq 0$, *and oracle access to a weight function* $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, *the value* $\mathrm{ed}^w_{\leq k}(X, Y)$ *can be computed in* $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ *time.*

Theorem (CKW, FOCS'23)

Conditioned on the All-Pairs Shortest-Paths Hypothesis, the above running time is optimal (up to subpolynomial factors) for $\sqrt{n} \leq k \leq n$.

Open Problems:

1. What is the true complexity of $\sqrt[3]{n} \leq k \leq \sqrt{n}$? Must be between $n + \sqrt{k^5}$ and $\sqrt{nk^3}$.
2. Is the problem easier for small (e.g., constant-sized) alphabets?

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\le n}$, a threshold $k \ge 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\ge 0}$, the value $\mathrm{ed}^w_{\le k}(X, Y)$ can be computed in $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

Theorem (CKW, FOCS'23)

Conditioned on the All-Pairs Shortest-Paths Hypothesis, the above running time is optimal (up to subpolynomial factors) for $\sqrt{n} \le k \le n$.

Open Problems:

1. What is the true complexity of $\sqrt[3]{n} \le k \le \sqrt{n}$? Must be between $n + \sqrt{k^5}$ and $\sqrt{nk^3}$.
2. Is the problem easier for small (e.g., constant-sized) alphabets?
3. Except for uniform weights, is there any easy class of weight functions?

Summary and Open Problems

Theorem (CKW, FOCS'23)

Given strings $X, Y \in \Sigma^{\leq n}$, a threshold $k \geq 0$, and oracle access to a weight function $w : (\Sigma \cup \{\varepsilon\})^2 \to \mathbb{R}_{\geq 0}$, the value $\mathrm{ed}^w_{\leq k}(X, Y)$ can be computed in $\widetilde{\mathcal{O}}(n + \sqrt{nk^3})$ time.

Theorem (CKW, FOCS'23)

Conditioned on the All-Pairs Shortest-Paths Hypothesis, the above running time is optimal (up to subpolynomial factors) for $\sqrt{n} \leq k \leq n$.

Open Problems:

1. What is the true complexity of $\sqrt[3]{n} \leq k \leq \sqrt{n}$? Must be between $n + \sqrt{k^5}$ and $\sqrt{nk^3}$.
2. Is the problem easier for small (e.g., constant-sized) alphabets?
3. Except for uniform weights, is there any easy class of weight functions?

Thank you for your attention!