MAX PLANCK INSTITUTE
FOR INFORMATICS

SIC Saarland Informatics Campus

Optimal Algorithms for Bounded Weighted Edit Distance

Alejandro Cassis
Saarland University
MPI-INF, SIC

Tomasz Kociumaka
MPI-INF, SIC

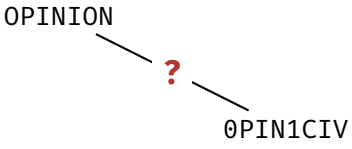
Philip Wellnitz
MPI-INF, SIC

Key Messages

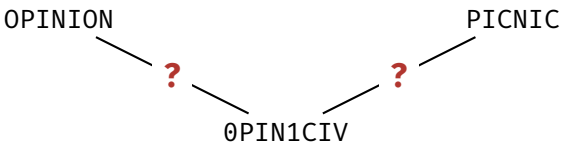
- Bounded Weighted Edit Distance is harder than Bounded Unweighted Edit Distance
- (Weighted) Edit Distance is Shortest Paths on Planar Graphs
- APSP-based Lower Bounds via Grid Construction and Dynamic Intermediate Problem

How similar are two strings X and Y?

How similar are two strings X and Y?



How similar are two strings X and Y?



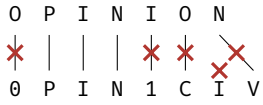
Edit Distance $\mathrm{ed}(X, Y)$

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

Edit Distance

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

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

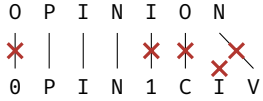


$\mathrm{ed}(\texttt{0PIN1CIV}, \texttt{OPINION}) = 5$

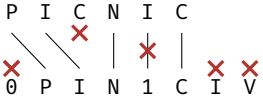
Edit Distance

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

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



$\mathrm{ed}(\texttt{0PIN1CIV}, \texttt{OPINION}) = 5$



$\mathrm{ed}(\texttt{0PIN1CIV}, \texttt{PICNIC}) = 5$

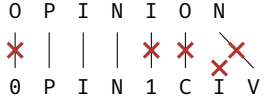
Weighted Edit Distance

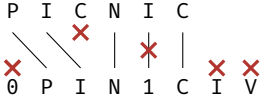
$\mathrm{ed}^w(X, Y)$

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

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

$w(0, \mathrm{O}) := 1 \quad w(1, \mathrm{I}) := 1 \quad w(\mathrm{C}, 0) := 1 \qquad w(*, *) := 2 \quad w(*, \varepsilon) := 1 \quad w(\varepsilon, *) := 10$





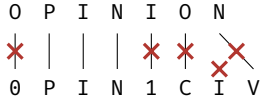
Weighted Edit Distance

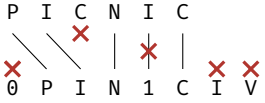
$\mathrm{ed}^w(X, Y)$

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

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

$w(0, O) := 1 \quad w(1, I) := 1 \quad w(C, O) := 1 \qquad w(*, *) := 2 \quad w(*, \varepsilon) := 1 \quad w(\varepsilon, *) := 10$





$\mathrm{ed}^w(\texttt{0PIN1CIV}, \texttt{OPINION}) = 6$

$\mathrm{ed}^w(\texttt{0PIN1CIV}, \texttt{PICNIC}) \leq 14$

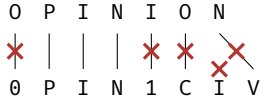
Weighted Edit Distance

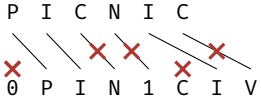
$\mathrm{ed}^w(X, Y)$

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

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

$w(0, O) := 1 \quad w(1, I) := 1 \quad w(C, O) := 1 \qquad w(*, *) := 2 \quad w(*, \varepsilon) := 1 \quad w(\varepsilon, *) := 10$





$\mathrm{ed}^w(0\mathrm{PIN1CIV}, \mathrm{OPINION}) = 6$

$\mathrm{ed}^w(0\mathrm{PIN1CIV}, \mathrm{PICNIC}) = 8$

How fast can we compute the (weighted) edit distance of two strings?

$\rightsquigarrow O(n^2)$ (you know this!)

How fast can we compute the (weighted) edit distance of two strings?

$\rightsquigarrow O(n^2)$ (you know this!)

What if the (weighted) edit distance is small (at most k)?

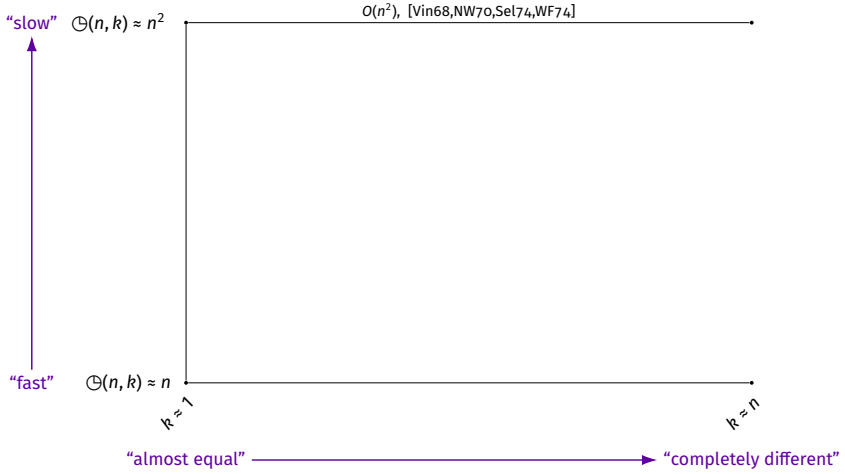
How fast can we compute the (weighted) edit distance of two strings?

$\rightsquigarrow O(n^2)$ (you know this!)

What if the (weighted) edit distance is small (at most k)?

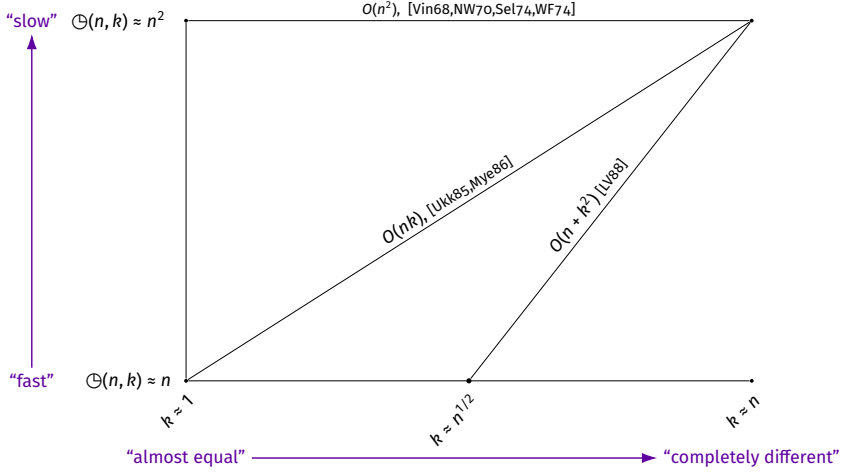
$\rightsquigarrow$ Bounded (Weighted) Edit Distance

Algorithms for (Bounded) Edit Distance



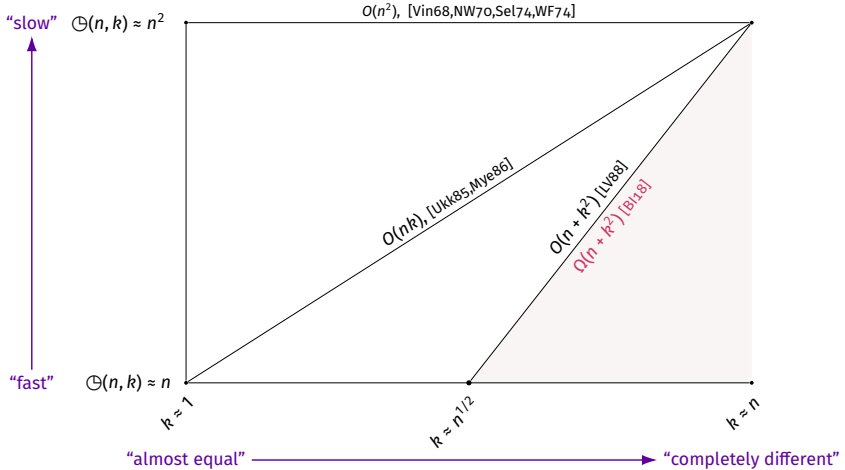
Existing algorithms for Edit Distance $\mathrm{ed}(X, Y)$, where $|X|, |Y| \leq n$

Algorithms for (Bounded) Edit Distance



Existing algorithms for Edit Distance $\mathrm{ed}(X, Y)$, where $|X|, |Y| \leq n$

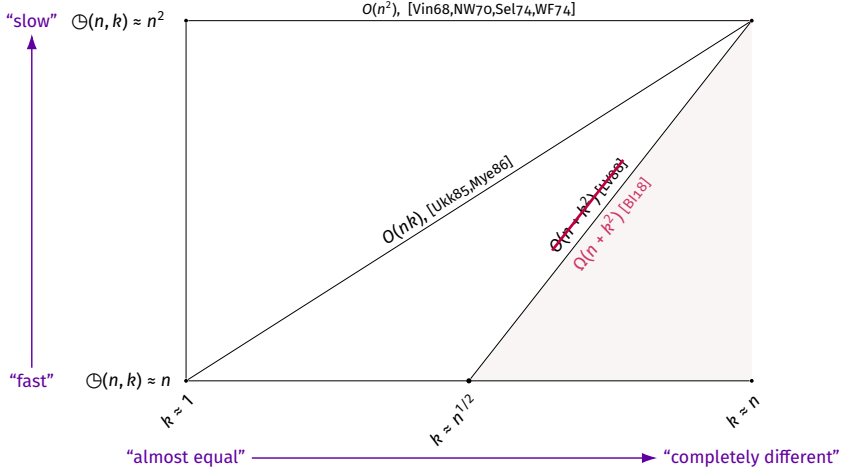
Algorithms for (Bounded) Edit Distance



Existing algorithms for Edit Distance $\mathrm{ed}(X, Y)$, where $|X|, |Y| \leq n$

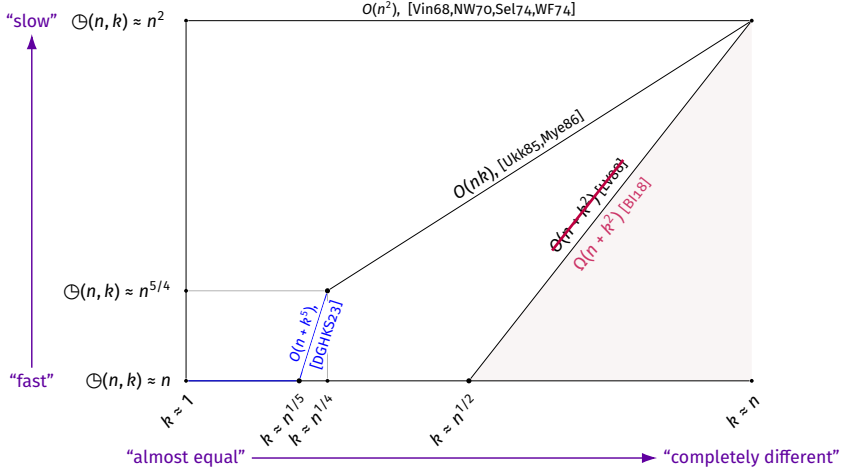
What about Weighted Edit Distance?

Algorithms for (Bounded) Weighted Edit Distance



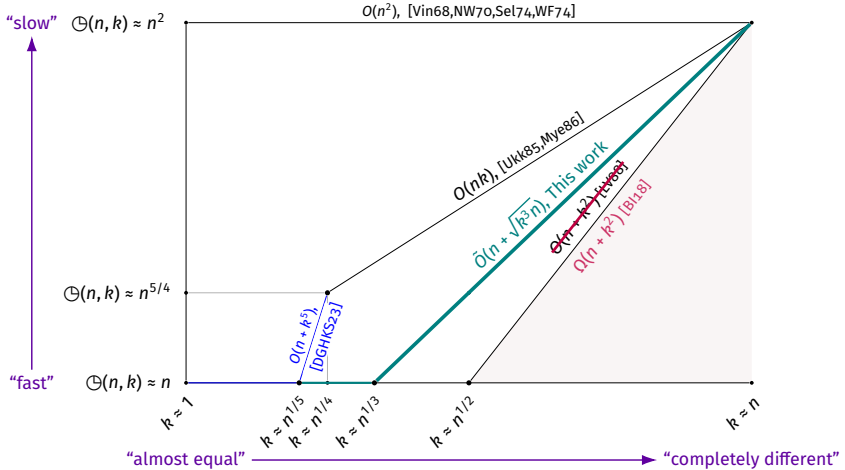
Existing algorithms for Weighted Edit Distance $\mathrm{ed}^w(X, Y)$, where $|X|, |Y| \leq n$

Algorithms for (Bounded) Weighted Edit Distance



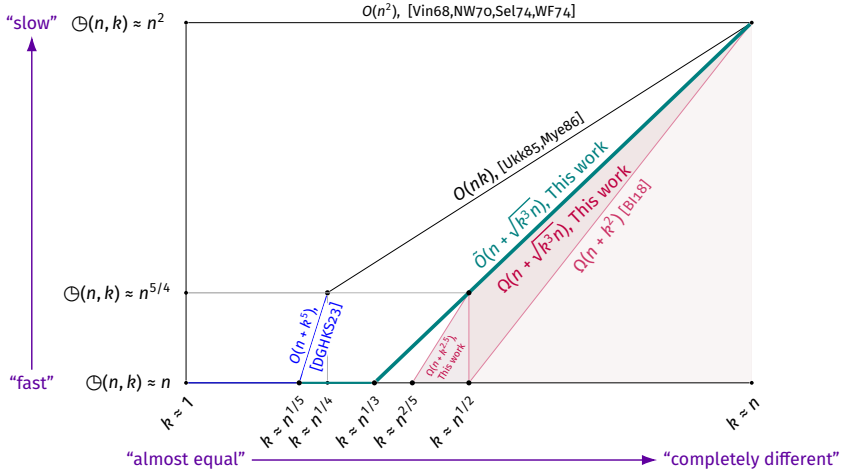
Existing algorithms for Weighted Edit Distance $\mathrm{ed}^w(X, Y)$, where $|X|, |Y| \leq n$

Algorithms for (Bounded) Weighted Edit Distance



Existing algorithms for Weighted Edit Distance $\mathrm{ed}^w(X, Y)$, where $|X|, |Y| \leq n$

Algorithms for (Bounded) Weighted Edit Distance



Existing algorithms for Weighted Edit Distance $\mathrm{ed}^w(X, Y)$, where $|X|, |Y| \le n$

Main Theorem 1 (Upper Bound)

Strings X, Y each of length at most n

Oracle access to (normalized) weight function w

Can compute $k = \mathrm{ed}^w(X, Y)$ in time $O(n + \sqrt{nk^3} \log^3 n)$

Main Theorem 1 (Upper Bound)

Strings X, Y each of length at most n
Oracle access to (normalized) weight function w
Can compute $k = \mathrm{ed}^w(X, Y)$ in time $O(n + \sqrt{nk^3} \log^3 n)$

Main Theorem 2 (Lower Bound)

Assuming the APSP Hypothesis and for $\sqrt{n} \leq k \leq n$,
Main Theorem 1 is tight (up to $n^{o(1)}$-factors)

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

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

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

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

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

- Justified Assumption: w is *normalized*, $w(x, y) \geq 1$ for all $x \neq y$.

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

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

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

- Justified Assumption: w is *normalized*, $w(x, y) \geq 1$ for all $x \neq y$.
  - $\rightsquigarrow$ Have (via $O(n + k^2)$ algo): Alignment $A : X \rightsquigarrow\!\!\!\!\rightarrow Y$ of unweighted cost $\leq k$

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

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

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

- Justified Assumption: w is *normalized*, $w(x, y) \geq 1$ for all $x \neq y$.
  - $\rightsquigarrow$ Have (via $O(n + k^2)$ algo): Alignment $A : X \rightsquigarrow\!\!\!\!\rightarrow Y$ of unweighted cost $\leq k$
  - $\rightsquigarrow$ A aligns edit-free most of X with Y [DGHKS23]

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

$\mathrm{ed}^{w}_{\le k}(X, Y)$

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

- Justified Assumption: w is *normalized*, $w(x, y) \ge 1$ for all $x \neq y$.
  - $\rightsquigarrow$ Have (via $O(n + k^2)$ algo): Alignment $A : X \rightsquigarrow\!\!\!\!\rightarrow Y$ of unweighted cost $\le k$
  - $\rightsquigarrow$ A aligns edit-free most of X with Y [DGHKS23]

Theorem (Universal Kernel)

[DGHKS23]

Can trim X and Y to length-$O(k^4)$ strings X', Y' with $\mathrm{ed}^{w}_{\le k}(X, Y) = \mathrm{ed}^{w}_{\le k}(X', Y')$.

Tool 0: Unweighted ED and [DGHKS23]-Kernel

Bounded Weighted Edit Distance

$\mathrm{ed}^{w}_{\le k}(X, Y)$

Min cost of transforming X to Y using character edits (if it is at most k), where:

- inserting y costs $w(\varepsilon, y)$;
- deleting x costs $w(x, \varepsilon)$;
- substituting x for y costs $w(x, y)$.

- Justified Assumption: w is *normalized*, $w(x, y) \ge 1$ for all $x \neq y$.
  - $\rightsquigarrow$ Have (via $O(n + k^2)$ algo): Alignment $A : X \rightsquigarrow\!\!\!\!\rightarrow Y$ of unweighted cost $\le k$
  - $\rightsquigarrow$ A aligns edit-free most of X with Y [DGHKS23]

Theorem (Universal Kernel)

[DGHKS23]

Can trim X and Y to length-$O(k^4)$ strings X', Y' with $\mathrm{ed}^{w}_{\le k}(X, Y) = \mathrm{ed}^{w}_{\le k}(X', Y')$.

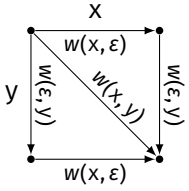
X' and Y' consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each

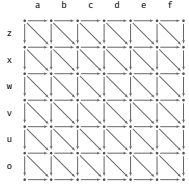
Tool 1: Alignment Graphs and Multiple-Source Shortest Path

- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each

Tool 1: Alignment Graphs and Multiple-Source Shortest Path

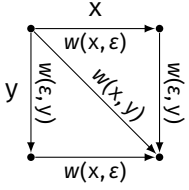
- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y

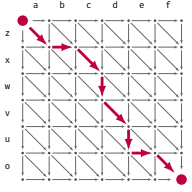




Tool 1: Alignment Graphs and Multiple-Source Shortest Path

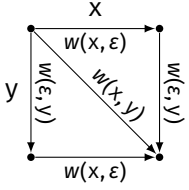
- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X, Y)$ is distance $(0, 0) \rightsquigarrow (|X|, |Y|)$

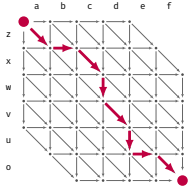




Tool 1: Alignment Graphs and Multiple-Source Shortest Path

- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X, Y)$ is distance $(0, 0)$ $\rightsquigarrow$ $(|X|, |Y|)$
- Idea^2: Trim AG to $O(k)$ diagonals $\rightsquigarrow$ $\mathrm{ed}^w_{\leq k}(X, Y)$ is distance $(0, 0)$ $\rightsquigarrow$ $(|X|, |Y|)$

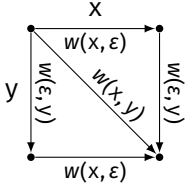


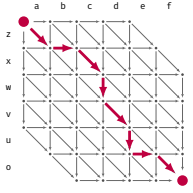


Tool 1: Alignment Graphs and Multiple-Source Shortest Path

- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X, Y)$ is distance $(0, 0)$ $\rightsquigarrow$ $(|X|, |Y|)$
- Idea^2: Trim AG to $O(k)$ diagonals $\rightsquigarrow$ $\mathrm{ed}^w_{\leq k}(X, Y)$ is distance $(0, 0)$ $\rightsquigarrow$ $(|X|, |Y|)$

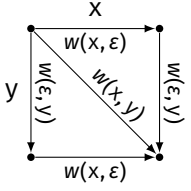
  $\rightsquigarrow$ AG has $O(k^5)$ vertices $\rightsquigarrow$ Dijkstra yields $\tilde{O}(k^5)$ algo

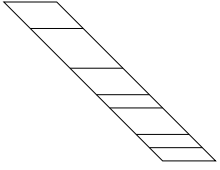




Tool 1: Alignment Graphs and Multiple-Source Shortest Path

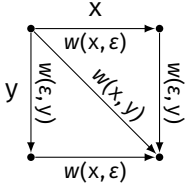
- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X,Y)$ is distance $(0,0)$ $\rightsquigarrow$ $(|X|,|Y|)$
- Idea^2: Trim AG to $O(k)$ diagonals $\rightsquigarrow$ $\mathrm{ed}^w_{\le k}(X,Y)$ is distance $(0,0)$ $\rightsquigarrow$ $(|X|,|Y|)$
- Idea^3: Split AG according to structure of X and Y
  Compute all b-to-b dist [Kleino5] + stitch together results ((min, +)-product [SMAWK87])

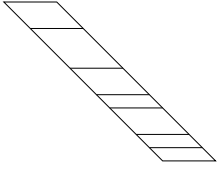




Tool 1: Alignment Graphs and Multiple-Source Shortest Path

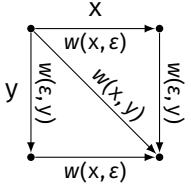
- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X, Y)$ is distance $(0, 0) \rightsquigarrow (|X|, |Y|)$
- Idea^2: Trim AG to $O(k)$ diagonals $\rightsquigarrow$ $\mathrm{ed}^w_{\le k}(X, Y)$ is distance $(0, 0) \rightsquigarrow (|X|, |Y|)$
- Idea^3: Split AG according to structure of X and Y
  Compute all b-to-b dist [Kleino5] + stitch together results ((min, +)-product [SMAWK87])
  $\rightsquigarrow k^2 \cdot \tilde{O}(k^3)$ for periodic pieces + $k^2 \cdot \tilde{O}(k^2)$ for stitching

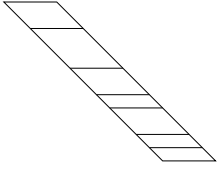




Tool 1: Alignment Graphs and Multiple-Source Shortest Path

- X and Y consist in $O(k^2)$ pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use *alignment graph* AG of X and Y $\rightsquigarrow$ $\mathrm{ed}^w(X,Y)$ is distance $(0,0)$ $\rightsquigarrow$ $(|X|,|Y|)$
- Idea^2: Trim AG to $O(k)$ diagonals $\rightsquigarrow$ $\mathrm{ed}^w_{\le k}(X,Y)$ is distance $(0,0)$ $\rightsquigarrow$ $(|X|,|Y|)$
- Idea^3: Split AG according to structure of X and Y
  Compute all b-to-b dist [Klein05] + stitch together results ((min, +)-product [SMAWK87]) for periodic pieces: fast exponentiation;
  $\rightsquigarrow$ $k^2 \cdot \tilde{O}(k^2)$ for periodic pieces + $k^2 \cdot \tilde{O}(k^2)$ for stitching





Tool 2: Divide and Conquer

- X and Y consist in $O(k^2)$ periodic pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use AG, trimmed to $O(k)$ diags $\rightsquigarrow \mathrm{ed}^{w}_{\leq k}(X, Y)$ is distance $(0, 0) \rightsquigarrow (|X|, |Y|)$
- Idea[3]: Compute all b-to-b dist for periodic pieces [Klein05] + fast exponentiation;
  stitch together results using min-plus product [SMAWK87]
- Idea[4]: Use Divide-and-Conquer to reduce number of periodic pieces to $O(k)$

Tool 2: Divide and Conquer

- X and Y consist in $O(k^2)$ periodic pieces of length $O(k^2)$ and with period $O(k)$ each
- Idea: Use AG, trimmed to $O(k)$ diags $\rightsquigarrow \mathrm{ed}^{w}_{\le k}(X,Y)$ is distance $(0,0) \rightsquigarrow (|X|,|Y|)$
- Idea[3]: Compute all b-to-b dist for periodic pieces [Klein05] + fast exponentiation;
  stitch together results using min-plus product [SMAWK87]
- Idea[4]: Use Divide-and-Conquer to reduce number of periodic pieces to $O(k)$
  $\rightsquigarrow k \cdot \tilde{O}(k^2)$ for periodic pieces (and padding) + $k \cdot \tilde{O}(k^2)$ for stitching

Tool 3: Compressibility instead of Periodicity

- X and Y consist in $O(k)$ pieces of length $O(k^2)$ each
- Idea: Use AG, trimmed to $O(k)$ diags $\rightsquigarrow \mathrm{ed}^{w}_{\leq k}(X, Y)$ is distance $(0, 0) \rightsquigarrow (|X|, |Y|)$
- Idea[3]: Compute all b-to-b dist for periodic pieces [Klein05] + fast exponentiation; stitch together results using min-plus product [SMAWK87]
- Idea[4]: Use Divide-and-Conquer to reduce number of periodic pieces to $O(k)$
- Idea[5]: Use tailor-made compressibility measure instead of periodicity + (w)ED algorithms for compressed strings

  $\rightsquigarrow \tilde{O}(n + \sqrt{k^3 n})$ time in total

Main Theorem 1 (Upper Bound) ✓

Strings X, Y each of length at most n
Oracle access to (normalized) weight function w
Can compute $k = \mathrm{ed}^w(X, Y)$ in time $O(n + \sqrt{nk^3} \log^3 n)$

Main Theorem 2 (Lower Bound)

Assuming the APSP Hypothesis and for $\sqrt{n} \leq k \leq n$,
Main Theorem 1 is tight (up to $n^{o(1)}$-factors)

Tool 0: APSP Hypothesis and Negative Triangle

All-Pairs Shortest Paths Hypothesis [VWW18]

There is no algorithm for APSP on n vertex graphs with running time $O(n^{3-\varepsilon})$.

Tool 0: APSP Hypothesis and Negative Triangle

All-Pairs Shortest Paths Hypothesis

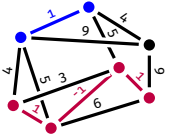
[VWW18]

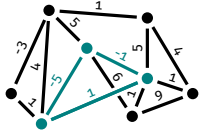
There is no algorithm for APSP on n vertex graphs with running time $O(n^{3-\varepsilon})$.

Negative Triangle

Check if n vertex graph has a negative triangle (3 vertices u, v, x with $w(u,v) + w(v,x) + w(u,x) < 0$)

APSP-H $\overset{\text{[VWW18]}}{\Longleftrightarrow}$ Negative Triangle





Tool 0: APSP Hypothesis and Negative Triangle

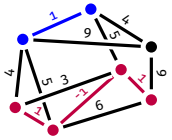
All-Pairs Shortest Paths Hypothesis [VWW18]

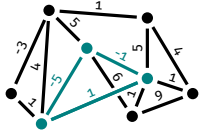
There is no algorithm for APSP on n vertex graphs with running time $O(n^{3-\varepsilon})$.

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of (min, +)-product $A \oplus B \oplus C$ is negative

APSP-H $\overset{\text{[VWW18]}}{\Longleftrightarrow}$ Negative Triangle $\overset{\text{[VWW18]}}{\Longleftrightarrow}$ MPP-Minimum





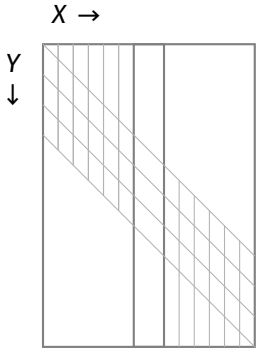
$$\min\left(\boxed{A} \oplus \boxed{B} \oplus \boxed{C}\right) < 0$$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min,+)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  $\rightsquigarrow$ encode matrix/vector as substitution costs



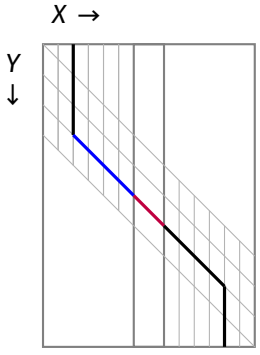
$\min_i A_{i,j} \oplus b_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min,+)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  $\rightsquigarrow$ encode matrix/vector as substitution costs



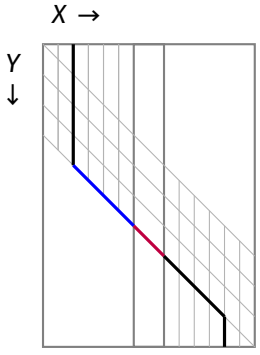
$\min_i A_{i,j} \oplus b_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min,+)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  $\rightsquigarrow$ encode matrix/vector as substitution costs



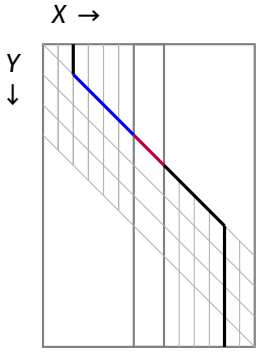
$\min_i A_{i,j} \oplus b_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min,+)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  $\rightsquigarrow$ encode matrix/vector as substitution costs



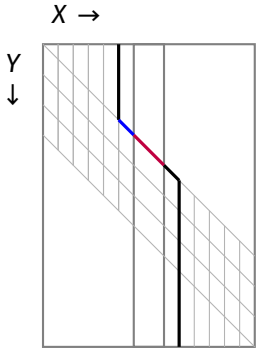
$\min_i A_{i,j} \oplus b_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min,+)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  - $\rightsquigarrow$ encode matrix/vector as substitution costs
  - $\rightsquigarrow$ use telescoping sums



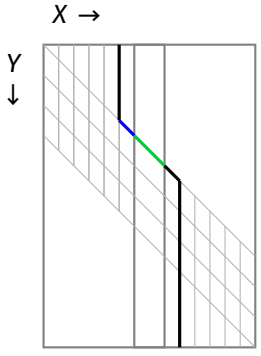
$\min_i A_{i,j} \oplus b_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of $(\min, +)$-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  - $\rightsquigarrow$ encode matrix/vector as substitution costs
  - $\rightsquigarrow$ use telescoping sums
- Step 2: Compute $A \oplus b'$ by replacing single character of X



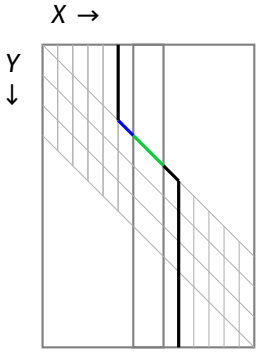
$\min_i A_{i,j} \oplus b'_i \approx \mathrm{ed}^w(X[j\,..\,|X|-j), Y)$

Tool 1: Matrix-Vector Min-Plus Multiplication via wED

Min-Plus Product Minimum

Given 3 matrices A, B, C, check if min diag entry of (min, +)-product $A \oplus B \oplus C$ is negative

- Step 1: Compute matrix-vector product $A \oplus b$ as weighted ED of two gadget strings
  - $\rightsquigarrow$ encode matrix/vector as substitution costs
  - $\rightsquigarrow$ use telescoping sums
- Step 2: Compute $A \oplus b'$ by replacing single character of X
  - $\rightsquigarrow$ Allows to compute $A \oplus B$; encode C similarly to B
  - $\rightsquigarrow$ Use selection gadget for $\mathrm{ed}^w(X, Y) \approx \min(A \oplus B \oplus C)$
  - $\rightsquigarrow$ Lower bound for dynamic problem



$\min_i A_{i,j} \oplus b'_i \approx \mathrm{ed}^w(X[j\,..\,|X| - j), Y)$

Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem

Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)

X_1 Y

X_2 Y

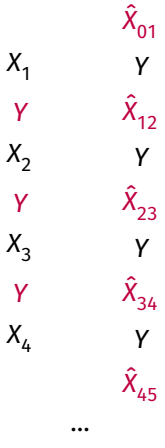
X_3 Y

X_4 Y

...

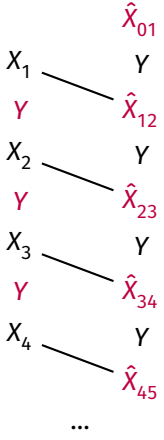
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$



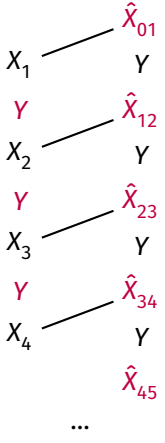
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost



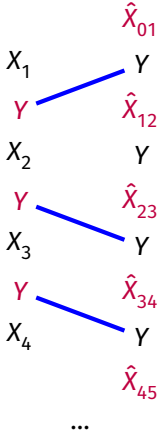
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost



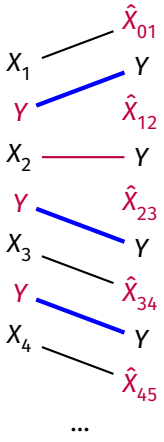
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost
- Step 5: Forbid inserting or deleting Y (simplification)
  $(w(\varepsilon, Y) := \infty, w(Y, \varepsilon) := \infty)$
  $\rightsquigarrow$ Have to align exactly one X_i to Y



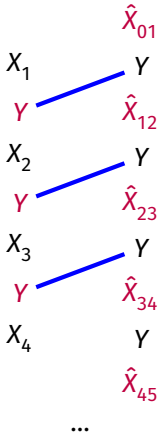
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost
- Step 5: Forbid inserting or deleting Y (simplification)
  $(w(\varepsilon, Y) := \infty, w(Y, \varepsilon) := \infty)$
  $\rightsquigarrow$ Have to align exactly one X_i to Y



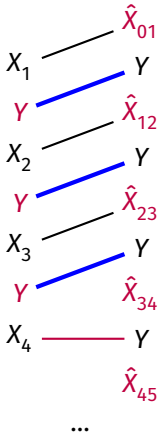
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost
- Step 5: Forbid inserting or deleting Y (simplification)
  $(w(\varepsilon, Y) := \infty, w(Y, \varepsilon) := \infty)$
  $\rightsquigarrow$ Have to align exactly one X_i to Y



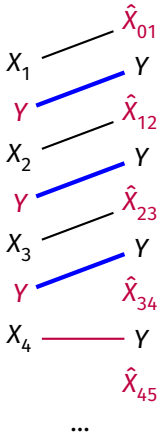
Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  $\rightsquigarrow$ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  $\rightsquigarrow$ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost
- Step 5: Forbid inserting or deleting Y (simplification)
  $(w(\varepsilon, Y) := \infty, w(Y, \varepsilon) := \infty)$
  $\rightsquigarrow$ Have to align exactly one X_i to Y



Tool 2: Minimum Gadget via Intermediate Strings

- Have: Lower bound for dynamic problem
  ⇝ turn into LB for static problem
- Step 3: Take snapshots (X_i, Y)
- Step 4: Create intermediate strings $(Y, \hat{X}_{(i-1)i})$
  ⇝ may align $\hat{X}_i$ with either X_i or X_{i+1} for the same cost
- Step 5: Forbid inserting or deleting Y (simplification)
  $(w(\varepsilon, Y) := \infty, w(Y, \varepsilon) := \infty)$
  ⇝ Have to align exactly one X_i to Y
  ⇝ Minimum gadget: $\mathrm{ed}^w(\hat{X}, \hat{Y}) \approx \min \mathrm{ed}^w(X_i, Y)$
  ⇝ Static LB



Main Theorem 1 (Upper Bound) ✓

Strings X, Y each of length at most n
Oracle access to (normalized) weight function w
Can compute $k = \mathrm{ed}^w(X, Y)$ in time $O(n + \sqrt{nk^3} \log^3 n)$

Main Theorem 2 (Lower Bound) ✓

Assuming the APSP Hypothesis and for $\sqrt{n} \leq k \leq n$,
Main Theorem 1 is tight (up to $n^{o(1)}$-factors)

Key Messages and Open Problems

Key Messages

- Bounded Weighted Edit Distance is harder than Bounded Unweighted Edit Distance
- (Weighted) Edit Distance is Shortest Paths on Planar Graphs
- APSP-based Lower Bounds via Grid Construction and Dynamic Intermediate Problem

Key Messages

- Bounded Weighted Edit Distance is harder than Bounded Unweighted Edit Distance
- (Weighted) Edit Distance is Shortest Paths on Planar Graphs
- APSP-based Lower Bounds via Grid Construction and Dynamic Intermediate Problem

Open Questions

- What is the true complexity of $n^{1/3} \leq k \leq n^{1/2}$?
  Must be between $n + k^{2.5}$ and $\sqrt{nk^3}$.
- Is the problem easier for small (constant-sized) alphabets?
- Is there any easy class of weight functions?

Start

End