Pattern Matching under Weighted Edit Distance

Panagiotis Charalampopoulos
King's College London
United Kingdom

Tomasz Kociumaka
Max Planck Institute
Saarbrücken, Germany

Philip Wellnitz
National Institute of Informatics
Tokyo, Japan

FOCS 2025

An Example

...

(TU Kaiserslautern).

(Simons Institute and UC Berkeley), (University of Salzburg), (University of Salzburg).

(University of Pennsylvania), (University of Pennsylvania), (University of Pennsylvania).

(Reichman University, Herzliya, Israel and Birkbeck, University of London), (UC Berkeley and Max Planck Institute for Informatics, SIC, Saarbrücken, Germany), (Max Planck Institute for Informatics, SIC, Saarbrücken, Germany).

(National Institute of Informatics).

(UniversitÃ© Paris CitÃ©, CNRS, IRIF, F-75013, Paris, France); (Universitat PolitÃ¨cnica de Catalunya)

...

(ETH Zurich); (Toyota Technological Institute at Chicago)

(Merton College, University of Oxford, United Kingdom); (Mathematical Institute, University of Bonn, Germany); (Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbr|cken, Germany)

(The Academic College of Tel Aviv-Yafo), (UC Berkeley), (Weizmann Institute of Science), (University of California San Diego).

(CISPA Helmholtz Center for Information Security, Saarbrücken); (Simon Fraser University); (Duke University); (CISPA Helmholtz Center for Information Security, Saarbrücken); (Max Planck Institute for Informatics, SIC, Saarbrücken)

...

(Alibaba Group); (Microsoft Research); (Washington University in St. Louis); (Columbia University); (University of Chicago)

(Stony Brook University); (Max Planck Institute for Informatics, Saarbruecken)

(Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, France); (University of Warsaw, Poland); (Saarland University and Max Planck Institute for Informatics, SaarbrÃ¼cken, Germany); (University of Warsaw, Poland)

(ETH ZÃ¼rich); (TU Berlin)

(Carnegie Mellon University)

...

An Example

...

(TU Kaiserslautern).

(Simons Institute and UC Berkeley), (University of Salzburg), (University of Salzburg).

(University of Pennsylvania), (University of Pennsylvania), (University of Pennsylvania).

(Reichman University, Herzliya, Israel and Birkbeck, University of London), (UC Berkeley and Max Planck Institute for Informatics, SIC, **Saarbrücken**, Germany), (Max Planck Institute for Informatics, SIC, **Saarbrücken**, Germany).

(National Institute of Informatics).

(UniversitÃ© Paris CitÃ©, CNRS, IRIF, F-75013, Paris, France); (Universitat PolitÃ¨cnica de Catalunya)

...

(ETH Zurich); (Toyota Technological Institute at Chicago)

(Merton College, University of Oxford, United Kingdom); (Mathematical Institute, University of Bonn, Germany); (Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbr|cken, Germany)

(The Academic College of Tel Aviv-Yafo), (UC Berkeley), (Weizmann Institute of Science), (University of California San Diego).

(CISPA Helmholtz Center for Information Security, **Saarbrücken**); (Simon Fraser University); (Duke University); (CISPA Helmholtz Center for Information Security, **Saarbrücken**); (Max Planck Institute for Informatics, SIC, **Saarbrücken**)

...

(Alibaba Group); (Microsoft Research); (Washington University in St. Louis); (Columbia University); (University of Chicago)

(Stony Brook University); (Max Planck Institute for Informatics, Saarbruecken)

(Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, France); (University of Warsaw, Poland); (Saarland University and Max Planck Institute for Informatics, SaarbrÃ¼cken, Germany); (University of Warsaw, Poland)

(ETH ZÃ¼rich); (TU Berlin)

(Carnegie Mellon University)

...

An Example

...

(TU Kaiserslautern).

(Simons Institute and UC Berkeley), (University of Salzburg), (University of Salzburg).

(University of Pennsylvania), (University of Pennsylvania), (University of Pennsylvania).

(Reichman University, Herzliya, Israel and Birkbeck, University of London), (UC Berkeley and Max Planck Institute for Informatics, SIC, **Saarbrücken**, Germany), (Max Planck Institute for Informatics, SIC, **Saarbrücken**, Germany).

(National Institute of Informatics).

(UniversitÃ© Paris CitÃ©, CNRS, IRIF, F-75013, Paris, France); (Universitat PolitÃ¨cnica de Catalunya)

...

...

(ETH Zurich); (Toyota Technological Institute at Chicago)

(Merton College, University of Oxford, United Kingdom); (Mathematical Institute, University of Bonn, Germany); (Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), **Saarbr|cken**, Germany)

(The Academic College of Tel Aviv-Yafo), (UC Berkeley), (Weizmann Institute of Science), (University of California San Diego).

(CISPA Helmholtz Center for Information Security, **Saarbrücken**); (Simon Fraser University); (Duke University); (CISPA Helmholtz Center for Information Security, **Saarbrücken**); (Max Planck Institute for Informatics, SIC, **Saarbrücken**)

...

...

(Alibaba Group); (Microsoft Research); (Washington University in St. Louis); (Columbia University); (University of Chicago)

(Stony Brook University); (Max Planck Institute for Informatics, **Saarbruecken**)

(Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, France); (University of Warsaw, Poland); (Saarland University and Max Planck Institute for Informatics, **SaarbrÃ¼cken**, Germany); (University of Warsaw, Poland)

(ETH ZÃ¼rich); (TU Berlin)

(Carnegie Mellon University)

...

Task: Find **Saarbrücken** in a text.

Task: Find **Saarbrücken** in a text.

Or Saarbruecken.

Task: Find **Saarbrücken** in a text.

Or Saarbruecken. Or Sarrebruck.

Task: Find **Saarbrücken** in a text.

Or Saarbruecken. Or Sarrebruck. Or Saarbrucken, Saarbr|cken, SaarbrÃ¼cken, ...

early 1980's

Approximate Pattern Matching

Given a text T, a pattern P, and a threshold k, identify the (starting positions of) substrings of T that are at **edit distance** of at most k to P.

early 1980's

Approximate Pattern Matching

Given a text T, a pattern P, and a threshold k, identify the (starting positions of) substrings of T that are at **edit distance** of at most k to P.

Edit distance: minimum number of insertions, deletions, or substitutions of single characters to transform one string into another string

early 1980's

Approximate Pattern Matching

Given a text T, a pattern P, and a threshold k, identify the (starting positions of) substrings of T that are at **edit distance** of at most k to P.

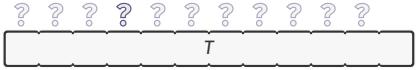
Applications

- **Text Retrieval**: Find occurrences of patterns or phrases in a text, accounting for misspellings or conversion errors
- **Signal Processing**: Find patterns in signals, accounting for transmission errors
- **Computational Biology**: Find specific patterns in DNA sequences, accounting for mutations or evolutionary alterations

early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.



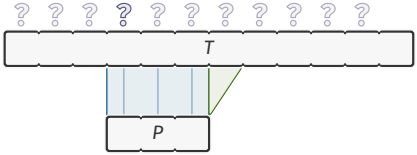
Focus: Obtain starting positions of occurrences

Approximate Pattern Matching

early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Focus: Obtain starting positions of occurrences

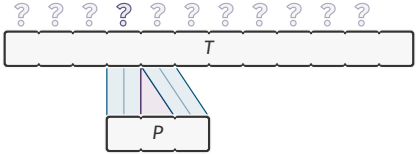


early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Focus: Obtain starting positions of occurrences

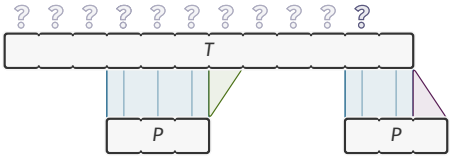


early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Focus: Obtain starting positions of occurrences



Approximate Pattern Matching

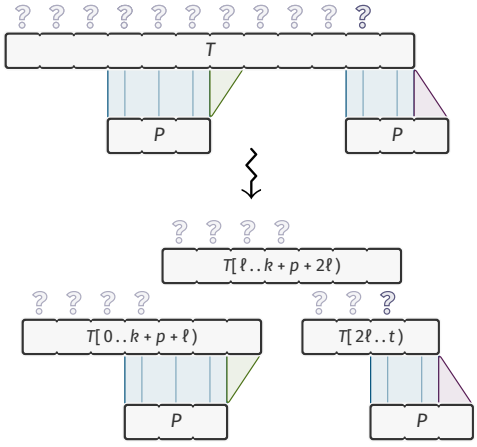
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Focus: Obtain starting positions of occurrences

"Standard Trick": write $t := |T|$, $p := |P|$
Split T into overlapping fragments of len $\ell + p + k$

$\rightsquigarrow O(t/\ell)$ instances,
each "responsible" for its first ℓ positions



Approximate Pattern Matching

early 1980's

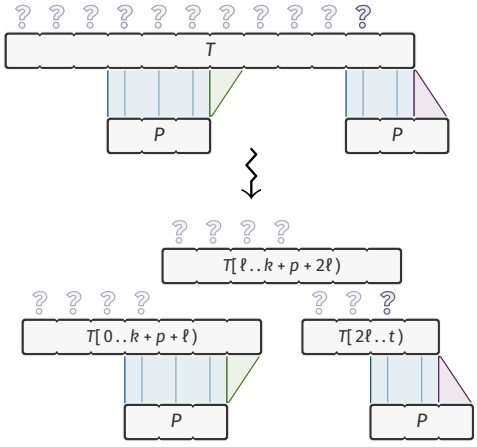
Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Focus: Obtain starting positions of occurrences

"Standard Trick": write $t := |T|$, $p := |P|$
Split T into overlapping fragments of len $\ell + p + k$

$\rightsquigarrow O(t/\ell)$ instances,
each "responsible" for its first ℓ positions

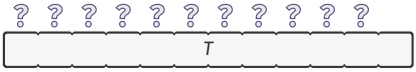
$\rightsquigarrow$ Useful special cases: $\ell = k$ and $\ell = 0.5\,p$



early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.



"Filter and Verify Paradigm"

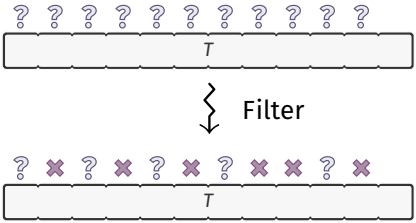
Approximate Pattern Matching

early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions



Approximate Pattern Matching

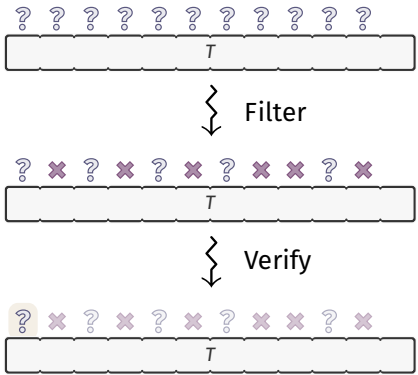
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

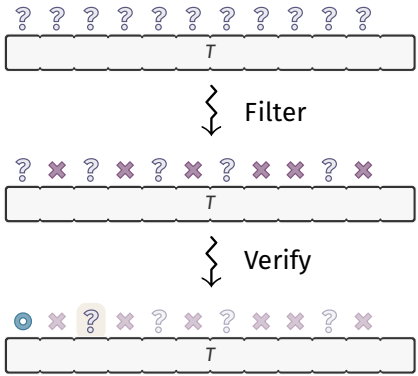
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

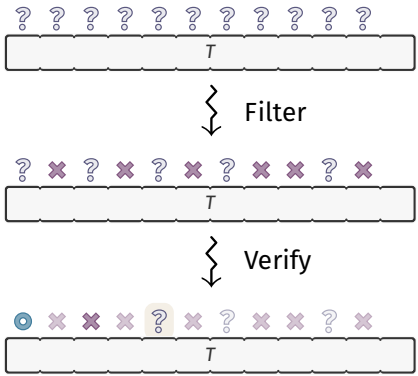
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

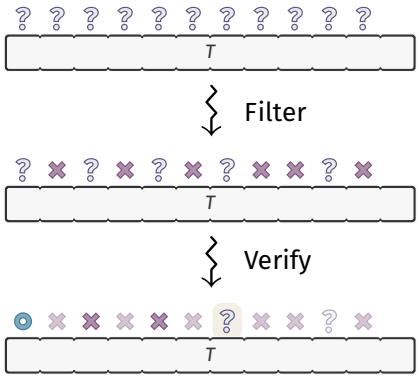
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

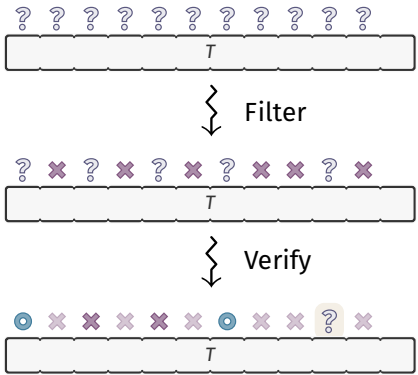
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

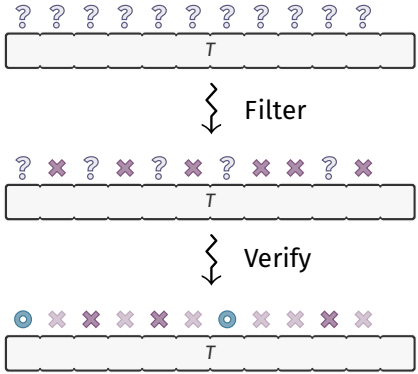
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2, Verify: (typically slow)
Check for occ at each remaining position



Approximate Pattern Matching

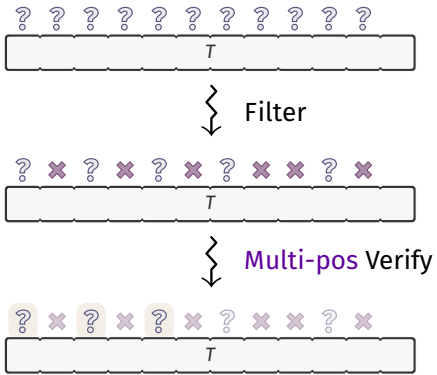
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2', Multi-pos Verify: (typically slow)
Check for occ at each remaining position, multiple positions at once



Approximate Pattern Matching

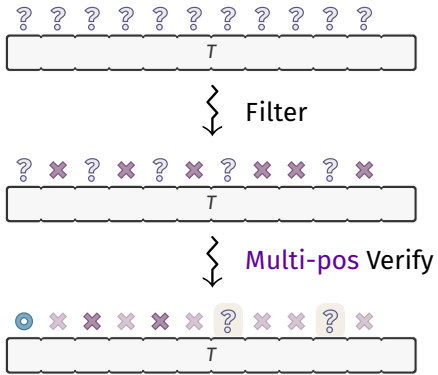
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2', Multi-pos Verify: (typically slow)
Check for occ at each remaining position, multiple positions at once



Approximate Pattern Matching

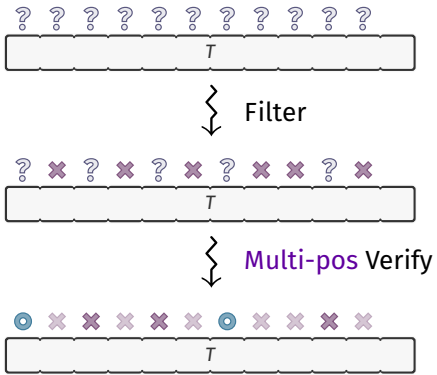
early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

"Filter and Verify Paradigm"

Step 1, Filter: (typically fast)
Compute (small) superset of starting positions

Step 2', Multi-pos Verify: (typically slow)
Check for occ at each remaining position, multiple positions at once



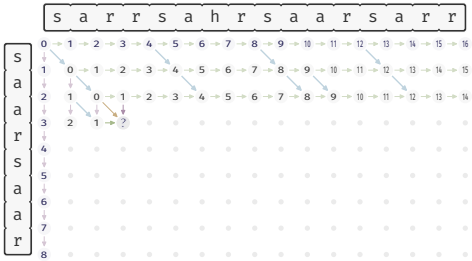
early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$	Verify pos 1 by 1 $\rightsquigarrow O(t^2p)$
[Sellers, 1980] and others	

$t := |T|, p := |P|$



$e_{i,0} = i$ (delete $T[0..i)$), $e_{0,j} = j$ (insert $P[0..j)$)

$$e_{i,j} = \min \begin{cases} e_{i-1,j} + 1 & \text{(del from } T\text{)} \\ e_{i,j-1} + 1 & \text{(ins in } T\text{)} \\ e_{i-1,j-1} + [T[i] \neq P[j]] & \text{(match/subst)} \end{cases}$$

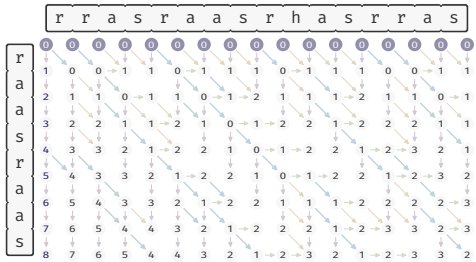
early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$	Verify pos 1 by 1 $\rightsquigarrow O(t^2 p)$
[Sellers, 1980] and others	Verify all pos at once $\rightsquigarrow O(tp)$

$t := |T|, p := |P|$



$e_{i,0} = 0$ (delete $T[0..i)$), $e_{0,j} = j$ (insert $P[0..j)$)

$$e_{i,j} = \min \begin{cases} e_{i-1,j} + 1 & \text{(del from } T\text{)} \\ e_{i,j-1} + 1 & \text{(ins in } T\text{)} \\ e_{i-1,j-1} + [T[i] \neq P[j]] & \text{(match/subst)} \end{cases}$$

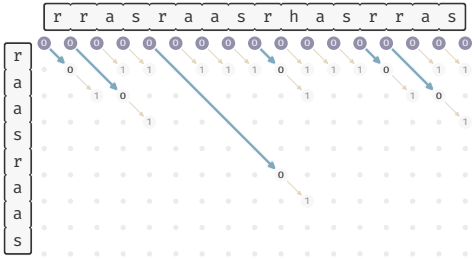
early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$ [Sellers, 1980] and others	Verify pos 1 by 1 $\rightsquigarrow O(t^2p)$ Verify all pos at once $\rightsquigarrow O(tp)$
Compute DP diagonals, jump over equal substr in $O(1)$ time $\rightsquigarrow O(t + k^2)$ [Landau, Vishkin'89]	Verify pos 1 by 1 $\rightsquigarrow O(tk^2)$ Verify $\ell \leq t$ pos at once $\rightsquigarrow O(t/\ell \cdot (\ell + k)k) = O(tk)$

$t := |T|, p := |P|$



Compute furthest pos on diag d reachable w/ $0, \ldots, k$ edits.
$\rightsquigarrow$ Jump over equal substrings in $O(1)$ time

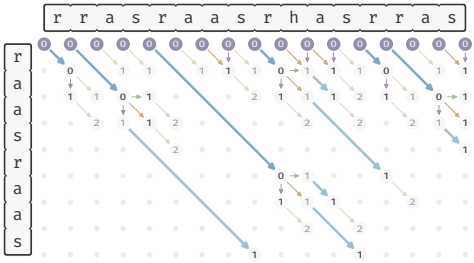
early 1980's

Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$ [Sellers, 1980] and others	Verify pos 1 by 1 $\rightsquigarrow O(t^2p)$ Verify all pos at once $\rightsquigarrow O(tp)$
Compute DP diagonals, jump over equal substr in $O(1)$ time $\rightsquigarrow O(t + k^2)$ [Landau, Vishkin'89]	Verify pos 1 by 1 $\rightsquigarrow O(tk^2)$ Verify $\ell \leq t$ pos at once $\rightsquigarrow O(t/\ell \cdot (\ell + k)k) = O(tk)$

$t := |T|, p := |P|$



Compute furthest pos on diag d reachable w/ $0, \ldots, k$ edits.
$\rightsquigarrow$ Jump over equal substrings in $O(1)$ time

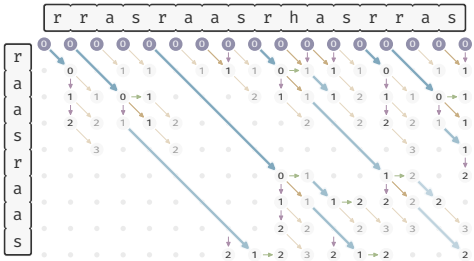
Approximate Pattern Matching

early 1980's

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$ [Sellers, 1980] and others	Verify pos 1 by 1 $\rightsquigarrow O(t^2p)$ Verify all pos at once $\rightsquigarrow O(tp)$
Compute DP diagonals, jump over equal substr in $O(1)$ time $\rightsquigarrow O(t + k^2)$ [Landau, Vishkin'89]	Verify pos 1 by 1 $\rightsquigarrow O(tk^2)$ Verify $\ell \leq t$ pos at once $\rightsquigarrow O(t/\ell \cdot (\ell + k)k) = O(tk)$

$t := |T|, p := |P|$



Compute furthest pos on diag d reachable w/ $0, \ldots, k$ edits.
$\rightsquigarrow$ Jump over equal substrings in $O(1)$ time

early 1980's

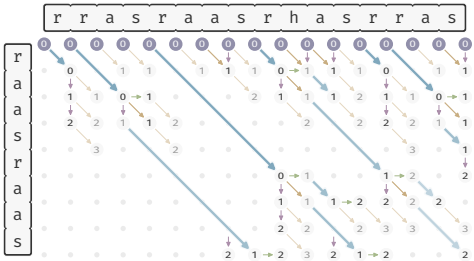
Approximate Pattern Matching

Given: text T, pattern P, threshold k; Find: (starting pos. of) substr. of T at edit distance $\leq k$ to P.

Edit dist/Verify Algorithm	APM Algorithm
Textbook DP $\rightsquigarrow O(tp)$ [Sellers, 1980] and others	Verify pos 1 by 1 $\rightsquigarrow O(t^2p)$ Verify all pos at once $\rightsquigarrow O(tp)$
Compute DP diagonals, jump over equal substr in $O(1)$ time $\rightsquigarrow O(t + k^2)$ [Landau, Vishkin'89]	Verify pos 1 by 1 $\rightsquigarrow O(tk^2)$ Verify $\ell \leq t$ pos at once $\rightsquigarrow O(t/\ell \cdot (\ell + k)k) = O(tk)$

$t := |T|, p := |P|$

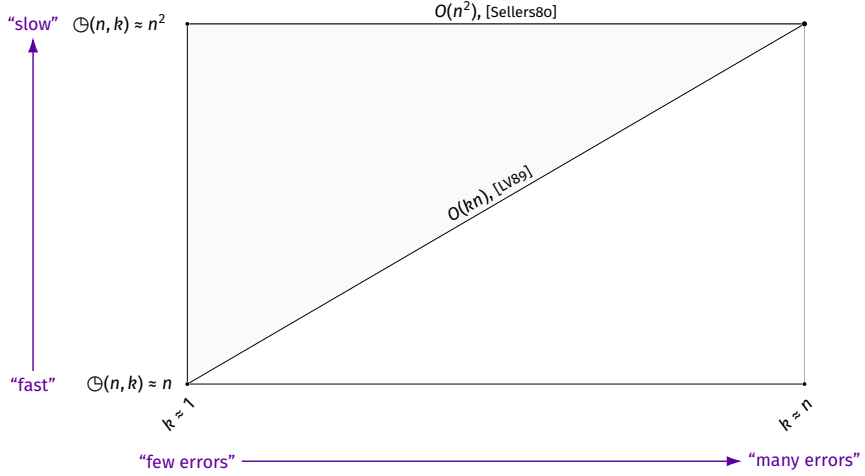
$\rightsquigarrow$ Didn't use filter...



Compute furthest pos on diag d reachable w/ $0, \ldots, k$ edits.

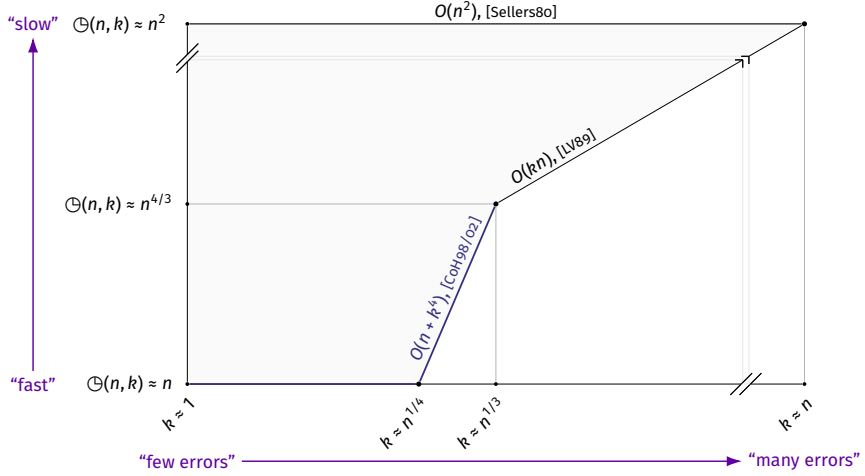
$\rightsquigarrow$ Jump over equal substrings in $O(1)$ time

State-of-the-Art Algorithms



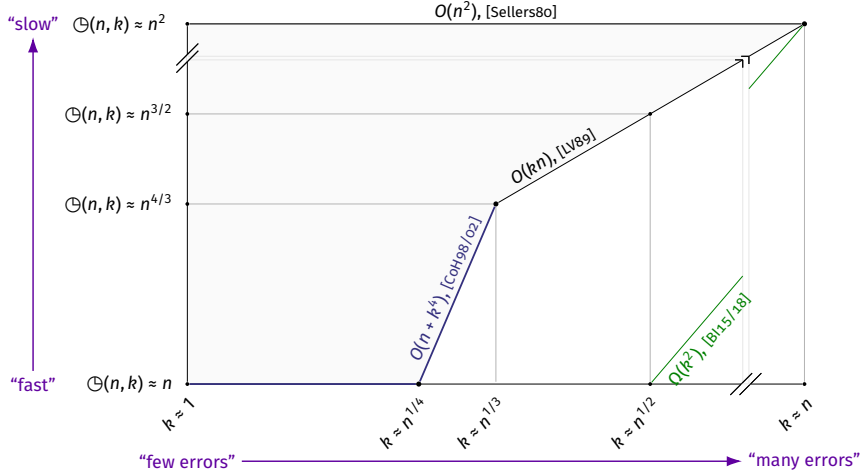
Existing algorithms for the case $n \approx t \approx 2p$

State-of-the-Art Algorithms



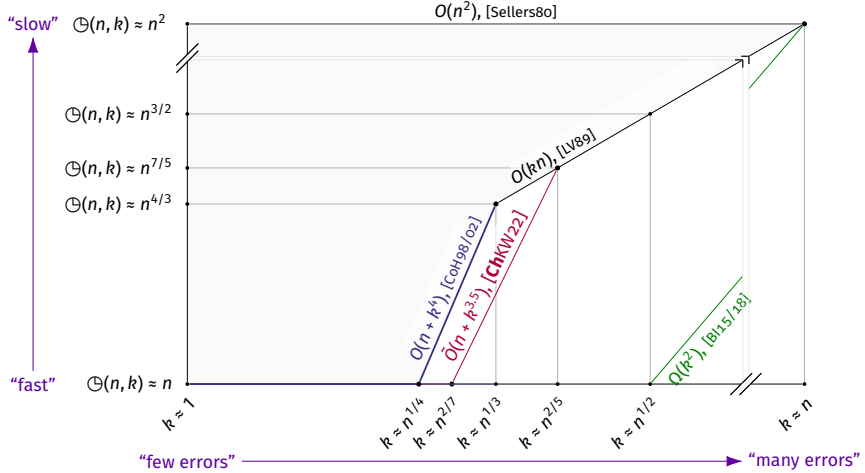
Existing algorithms for the case $n \approx t \approx 2p$

State-of-the-Art Algorithms



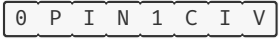
Existing algorithms for the case $n \approx t \approx 2p$

State-of-the-Art Algorithms

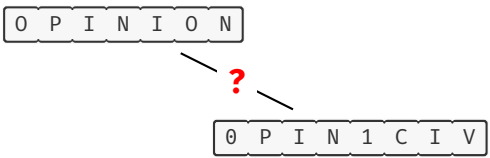


Existing algorithms for the case $n \approx t \approx 2p$

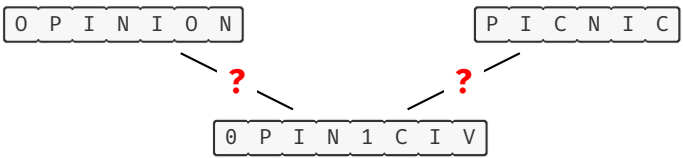
How similar are two strings X and Y?



How similar are two strings X and Y?



How similar are two strings X and Y?



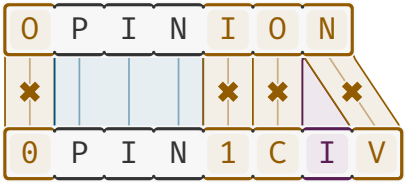
Edit Distance ED(X, Y)

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

Edit Distance

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

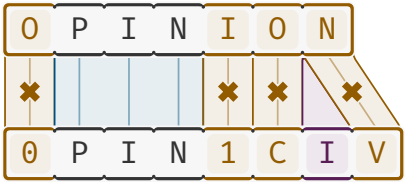
Min number of character insertions, deletions, and substitutions that transform X to Y



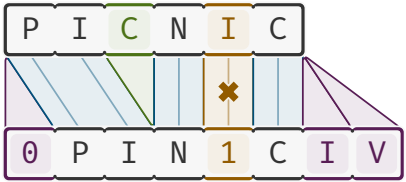
$\mathrm{ED}(\texttt{0PIN1CIV}, \texttt{OPINION}) = 5$

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

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



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



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

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

Weighted Edit Distance

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 $w(x, y)$.

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

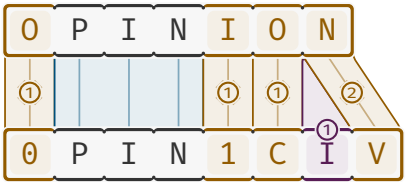
Weighted Edit Distance

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

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 $w(x, y)$.

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



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

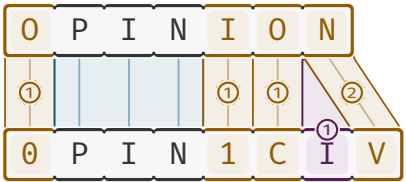
Weighted Edit Distance

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

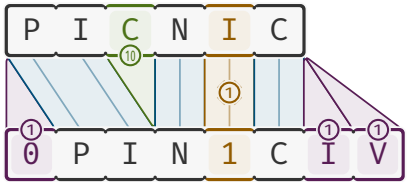
Min cost of transforming X to Y using character edits, where:

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

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



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



$\mathrm{ED}^w(\mathtt{0PIN1CIV}, \mathtt{PICNIC}) \leq 14$

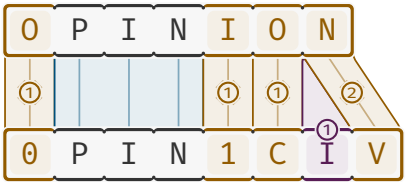
Weighted Edit Distance

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

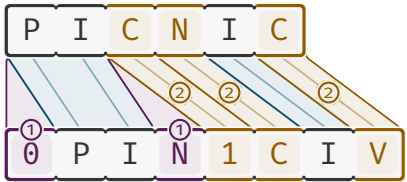
Min cost of transforming X to Y using character edits, where:

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

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



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



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

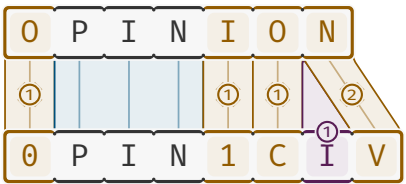
Weighted Edit Distance

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

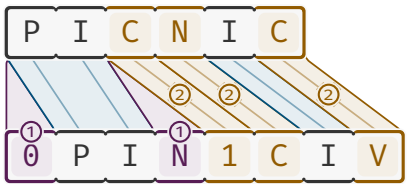
Min cost of transforming X to Y using character edits, where:

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

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



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



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

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

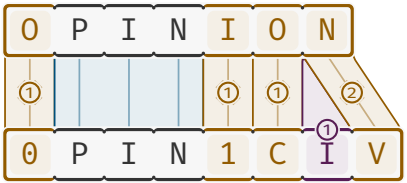
Weighted Edit Distance

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

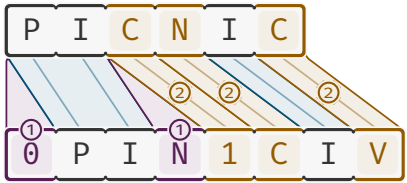
Min cost of transforming X to Y using character edits, where:

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

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



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



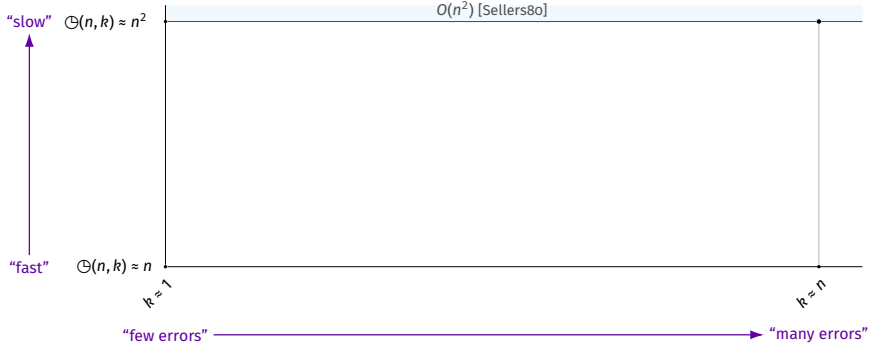
$\mathrm{ED}^w(\mathtt{0PIN1CIV}, \mathtt{PICNIC}) = 8$

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

Otherwise: could scale weights and parameterizing the running time by k does not make sense.

Weighted Approximate Pattern Matching

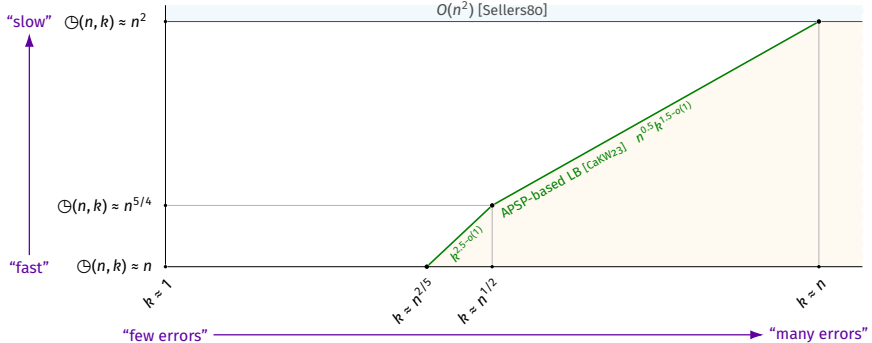
Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.



Existing algorithms for the case $n \approx t \approx 2p$

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.



Existing algorithms for the case $n \approx t \approx 2p$

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Some (unsatisfactory) upper bounds of the forms $t \cdot k^{O(1)}$ and $t \cdot \sqrt{p} \cdot k^{O(1)}$ follow from the state-of-the-art algorithm for weighted edit distance computation [Cassis, Kociumaka, Wellnitz'23].

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Some (unsatisfactory) upper bounds of the forms $t \cdot k^{O(1)}$ and $t \cdot \sqrt{p} \cdot k^{O(1)}$ follow from the state-of-the-art algorithm for weighted edit distance computation [Cassis, Kociumaka, Wellnitz'23].

How much harder is the weighted variant?

Weighted Approximate Pattern Matching

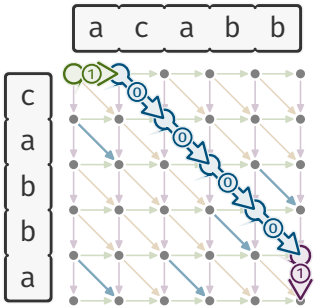
Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Some (unsatisfactory) upper bounds of the forms $t \cdot k^{O(1)}$ and $t \cdot \sqrt{p} \cdot k^{O(1)}$ follow from the state-of-the-art algorithm for weighted edit distance computation [Cassis, Kociumaka, Wellnitz'23].

How much harder is the weighted variant?

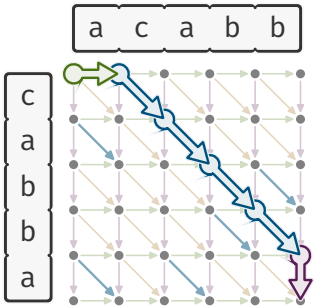
We nearly-match the best known upper bounds for the unweighted setting for a large range of parameters.

Main Challenge: Can't be Greedy



In the unweighted setting, when we see a long match, we greedily take it. This is the basis of the $O(kt)$-time algorithm we saw before.

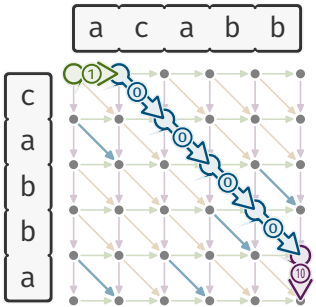
Main Challenge: Can't be Greedy



In the unweighted setting, when we see a long match, we greedily take it. This is the basis of the $O(kt)$-time algorithm we saw before.

What about in the weighted setting?

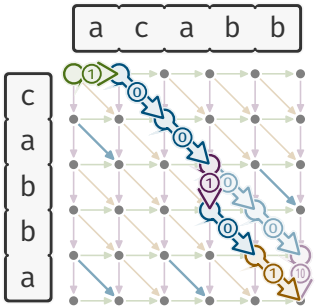
Main Challenge: Can't be Greedy



In the unweighted setting, when we see a long match, we greedily take it. This is the basis of the $O(kt)$-time algorithm we saw before.

What about in the weighted setting?

Main Challenge: Can't be Greedy



In the unweighted setting, when we see a long match, we greedily take it. This is the basis of the $O(kt)$-time algorithm we saw before.

What about in the weighted setting? A disaster.

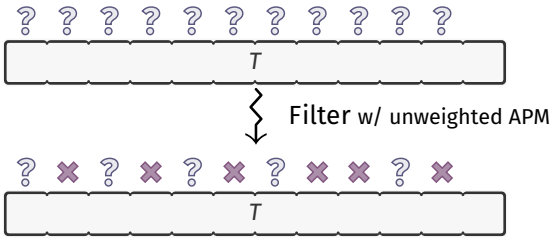
An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)

An $\tilde{O}(kt)$-time Algorithm for Weighted APM

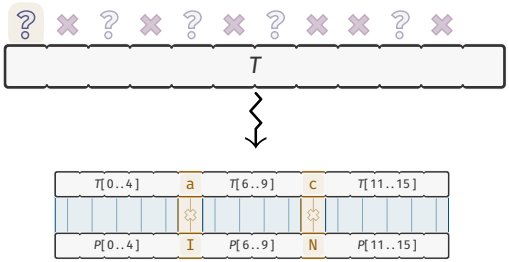
Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)

$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter



Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment $P \rightsquigarrow\!\!\!\rightarrow T[\,a\,..\,b\,]$ that matches exactly most of P

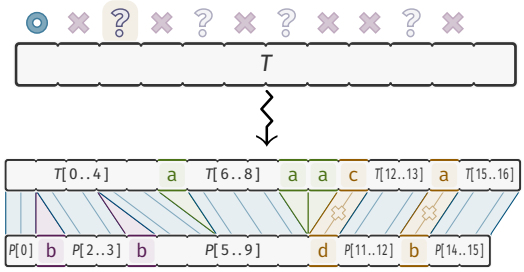


An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment
$P \rightsquigarrow T[\,a\,..\,b\,]$ that matches exactly most of P

All such alignments are very similar to each other
But how to exploit this fact?

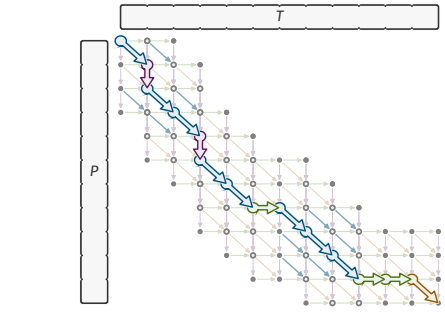


An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment
$P \rightsquigarrow\!\!\!\!\rightarrow T[\,a\,..\,b\,]$ that matches exactly most of P

All such alignments are very similar to each other
But how to exploit this fact?
$\rightsquigarrow$ Cast as shortest path prob in alignment graphs

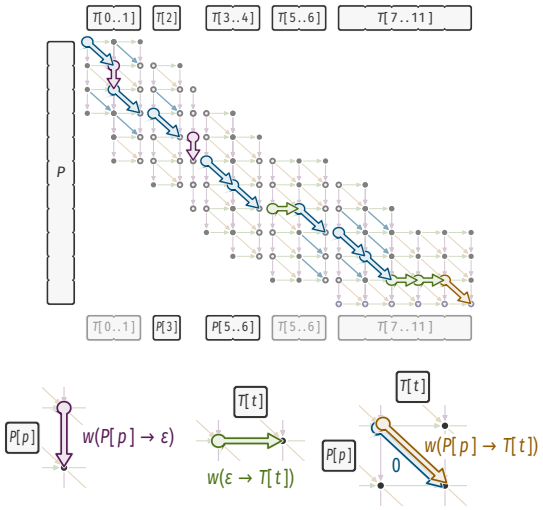


An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment $P \rightsquigarrow\!\!\!\gg T[\,a\,..\,b\,]$ that matches exactly most of P

All such alignments are very similar to each other
But how to exploit this fact?
$\rightsquigarrow$ Cast as shortest path prob in alignment graphs
$\rightsquigarrow$ Copy matched parts from (trivial) align. $P \rightsquigarrow\!\!\!\gg P$



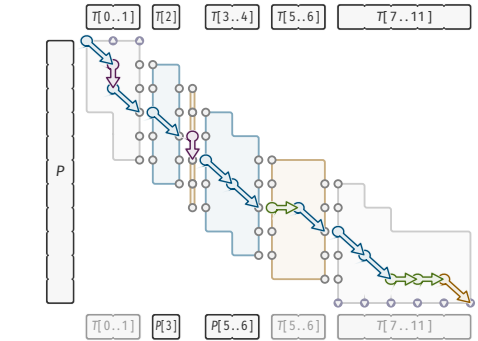
An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment $P \rightsquigarrow\!\!\!\gg T[\,a\,..\,b\,]$ that matches exactly most of P

All such alignments are very similar to each other
But how to exploit this fact?
$\rightsquigarrow$ Cast as shortest path prob in alignment graphs
$\rightsquigarrow$ Copy matched parts from (trivial) align. $P \rightsquigarrow\!\!\!\gg P$

Comp. bound-to-bound dist for each block [Klein'05];
then combine using fast (min, +)-product [SMAWK]



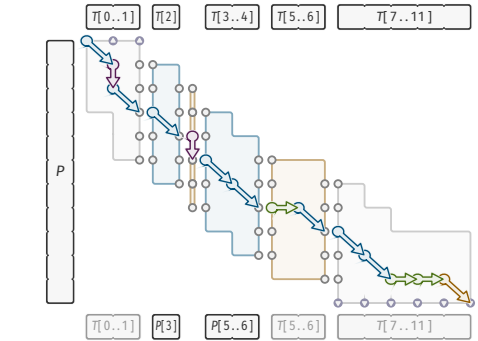
An $\tilde{O}(kt)$-time Algorithm for Weighted APM

Observation: $\mathrm{ED}^w(X, Y) \geq \mathrm{ED}(X, Y)$ (by norm.)
$\rightsquigarrow$ Use unweighted $O(kn)$-time APM as Filter

If $\mathrm{ED}(P, T[\,a\,..\,b\,]) \leq k$, also get cheap alignment
$P \rightsquigarrow\!\!\!\gg T[\,a\,..\,b\,]$ that matches exactly most of P

All such alignments are very similar to each other
But how to exploit this fact?
$\rightsquigarrow$ Cast as shortest path prob in alignment graphs
$\rightsquigarrow$ Copy matched parts from (trivial) align. $P \rightsquigarrow\!\!\!\gg P$

Comp. bound-to-bound dist for each block [Klein'05];
then combine using fast (min, +)-product [SMAWK]
$\rightsquigarrow$ Verify $\leq k$ pos in $\tilde{O}(k^2)$ after $\tilde{O}(kp)$ prep.
$\rightsquigarrow$ $\tilde{O}(kt)$ total



Summary of our Results

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Main Theorem

Weighted APM is in time $\tilde{O}(kt)$.

Main Theorem

Weighted APM is in time $\tilde{O}(t + k^4 \cdot t/p)$.

Summary of our Results

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Main Theorem

Weighted APM is in time $\tilde{O}(kt)$.

Main Theorem

Weighted APM is in time $\tilde{O}(t + k^4 \cdot t/p)$.

The latter result is general: compressed strings, dynamic strings, quantum algorithm, etc.

Summary of our Results

Weighted Approximate Pattern Matching

Given: T, P, k, oracle to w; Find: (starting pos. of) substr. of T at weighted ED $\leq k$ to P.

Main Theorem

Weighted APM is in time $\tilde{O}(kt)$.

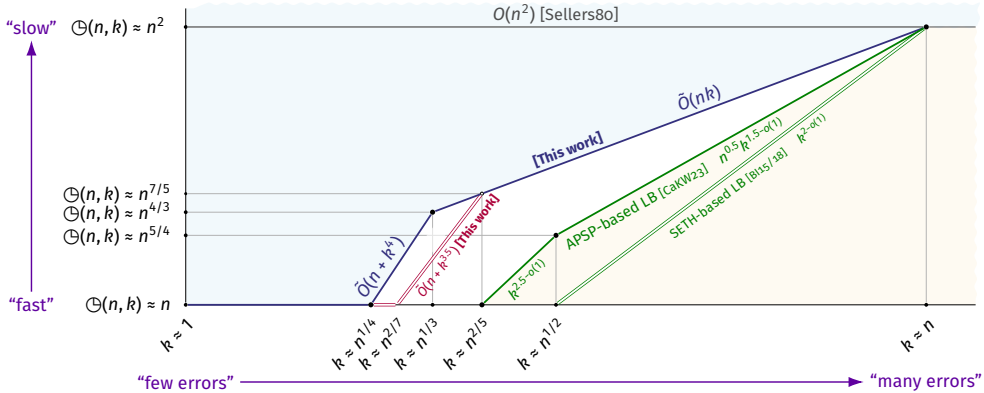
Main Theorem

Weighted APM is in time $\tilde{O}(t + k^4 \cdot t/p)$.

The latter result is general: compressed strings, dynamic strings, quantum algorithm, etc.

We can do better when the weights are small integers, matching the state of the art for the unweighted setting.

- Close gaps between UBs and LBs; there is a combinatorial barrier in improving the $\tilde{O}(t + k^4 \cdot t/p)$-time algorithm
- Report substrings instead of starting positions; we can do $\tilde{O}(k^2 t)$



Thank you!

Full version: https://arxiv.org/abs/2510.17752