Hardness of Tree Edit Distance and Friends

Bingbing Hu[1] **Jakob Nogler**[2] Barna Saha[1]

[1]UC San Diego

[2]MIT

(String) Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

(String) Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

1. Substitute a character c with c' with cost $\delta(c, c')$

`abcxef` ⟶ `abcyef`

(String) Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

1. Substitute a character c with c' with cost $\delta(c, c')$

   `abcxef` ⟶ `abcyef`

2. Delete a character c with cost $\delta(c, \varepsilon)$

   `abcdef` ⟶ `abdef`

(String) Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

1. Substitute a character c with c' with cost $\delta(c, c')$

   `abcxef` ⟶ `abcyef`

2. Delete a character c with cost $\delta(c, \varepsilon)$

   `abcdef` ⟶ `abdef`

3. Insert a character c with cost $\delta(\varepsilon, c)$

   `abcef` ⟶ `abcxef`

(String) Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

1. Substitute a character c with c' with cost $\delta(c, c')$

   `abcxef` ⟶ `abcyef`

2. Delete a character c with cost $\delta(c, \varepsilon)$

   `abcdef` ⟶ `abdef`

3. Insert a character c with cost $\delta(\varepsilon, c)$

   `abcef` ⟶ `abcxef`

"Unweighted" (String) Edit Distance: all costs are one

(String) Edit Distance Algorithms

References	Time	Remarks
Vin68, NW70, Sel74, WF74	$\mathcal{O}(n^2)$	textbook algorithm
MP80	$\mathcal{O}(n^2/\log^2 n)$	small alphabets and integer weights only
BF05	$\mathcal{O}(n^2 \log\log n/\log^2 n)$	small integer weights only
BI18	$\Omega(n^{2-o(1)})$	under SETH, already for unweighted

Dynamic (String) Edit Distance

Strings undergo updates (insertion/deletions and substitutions), and we need to maintain their edit distance.

$$\text{ed}(\ \texttt{port}\ ,\ \texttt{arts}\ ) = 3$$

Dynamic (String) Edit Distance

Strings undergo updates (insertion/deletions and substitutions), and we need to maintain their edit distance.

$$\text{ed}(\texttt{part}, \texttt{arts}) = 2$$

Dynamic (String) Edit Distance

Strings undergo updates (insertion/deletions and substitutions), and we need to maintain their edit distance.

$$\mathrm{ed}(\texttt{ part }, \texttt{ art }) = 1$$

Dynamic (String) Edit Distance

Strings undergo updates (insertion/deletions and substitutions), and we need to maintain their edit distance.

$$\mathrm{ed}(\,\texttt{part}\,,\ \texttt{art}\,) = 1$$

Question: Can we do better than recomputing from scratch?

Dynamic (String) Edit Distance

Strings undergo updates (insertion/deletions and substitutions), and we need to maintain their edit distance.

$$\text{ed}(\texttt{ part },\ \texttt{ art }) = 1$$

Question: Can we do better than recomputing from scratch?

References	**Update Time**	**Remarks**
CKM20	$\widetilde{\mathcal{O}}(n)$	unweighted
CKM20	$\widetilde{\mathcal{O}}(n \cdot \min(\sqrt{n}, W))$	uniform weights, W is maximum weight
GK25	$\widetilde{\mathcal{O}}(nW)$	arbitrary weights
CKW23	$\Omega(n^{1.5-o(1)})$	under APSP, large W

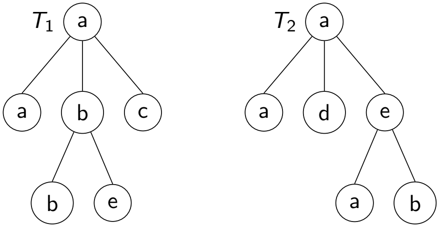
This Paper: Generalization of (String) Edit Distance in the Dynamic Setting

Tree Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ Generalization on Trees

Tree Edit Distance Problem (TED)
Input: Two rooted, labeled, left-to-right-ordered trees T_1, T_2 and a cost function δ.
Output: Cheapest transformation of T_1 into T_2 using deletion, insertions and substitutions.

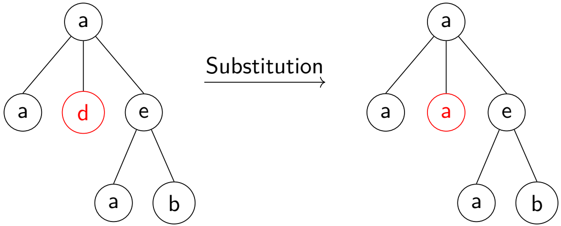


Tree Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ Generalization on Trees

Tree Edit Distance Problem (TED)
Input: Two rooted, labeled, left-to-right-ordered trees T_1, T_2 and a cost function δ.
Output: Cheapest transformation of T_1 into T_2 using deletion, insertions and substitutions.

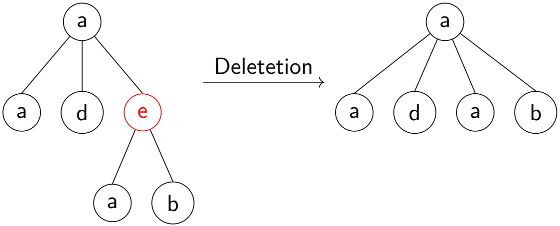


Tree Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ Generalization on Trees

Tree Edit Distance Problem (TED)
Input: Two rooted, labeled, left-to-right-ordered trees T_1, T_2 and a cost function δ.
Output: Cheapest transformation of T_1 into T_2 using deletion, insertions and substitutions.

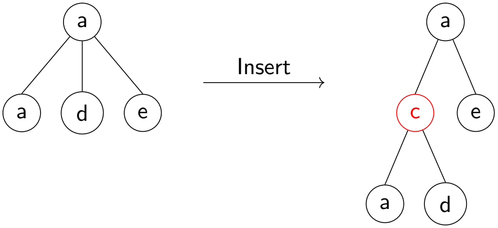


Tree Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ Generalization on Trees

Tree Edit Distance Problem (TED)
Input: Two rooted, labeled, left-to-right-ordered trees T_1, T_2 and a cost function δ.
Output: Cheapest transformation of T_1 into T_2 using deletion, insertions and substitutions.



Algorithms for Tree Edit Distance

Reference	Complexity	Remarks
Tai79	$\mathcal{O}(n^6)$	weighted
SZ89	$\mathcal{O}(n^4)$	weighted
Klein98	$\mathcal{O}(n^3 \log n)$	weighted
DMRW10	$\mathcal{O}(n^3)$	weighted
BGMW20	no $\mathcal{O}(n^{3-\varepsilon})$ algo under APSP	weighted
NPSVWXY25	$n^3/2^{\Omega(\sqrt{\log n})}$	weighted
Mao22	$\mathcal{O}(n^{2.9546})$	unweighted
Dürr23	$\mathcal{O}(n^{2.9148})$	unweighted
NPSVWXY25	$\mathcal{O}(n^{2.687})$	unweighted

Algorithms for Tree Edit Distance

Reference	Complexity	Remarks
Tai79	$\mathcal{O}(n^6)$	weighted
SZ89	$\mathcal{O}(n^4)$	weighted
Klein98	$\mathcal{O}(n^3 \log n)$	weighted
DMRW10	$\mathcal{O}(n^3)$	weighted
BGMW20	no $\mathcal{O}(n^{3-\varepsilon})$ algo under APSP	weighted
NPSVWXY25	$n^3/2^{\Omega(\sqrt{\log n})}$	weighted
Mao22	$\mathcal{O}(n^{2.9546})$	unweighted
Dürr23	$\mathcal{O}(n^{2.9148})$	unweighted
NPSVWXY25	$\mathcal{O}(n^{2.687})$	unweighted

Question 1: How about *Dynamic* Tree Edit Distance?

Algorithms for Tree Edit Distance

Reference	Complexity	Remarks
Tai79	$\mathcal{O}(n^6)$	weighted
SZ89	$\mathcal{O}(n^4)$	weighted
Klein98	$\mathcal{O}(n^3 \log n)$	weighted
DMRW10	$\mathcal{O}(n^3)$	weighted
BGMW20	no $\mathcal{O}(n^{3-\varepsilon})$ algo under APSP	weighted
NPSVWXY25	$n^3/2^{\Omega(\sqrt{\log n})}$	weighted
Mao22	$\mathcal{O}(n^{2.9546})$	unweighted
Dürr23	$\mathcal{O}(n^{2.9148})$	unweighted
NPSVWXY25	$\mathcal{O}(n^{2.687})$	unweighted

Question 1: How about *Dynamic* Tree Edit Distance?

Question 2: Is Unweighted Edit Distance on trees harder than on strings?

Dyck Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ About Balanced Sequence of Parenthesis

Dyck Edit Distance Problem
Input: A string S over a set of parenthesis
Output: Closest S' to S w.r.t. edit distance such that S' is balanced

Dyck Edit Distance

(String) Edit Distance Problem
Input: Two strings S_1, S_2 and a cost function δ.
Output: Cheapest transformation of S_1 into S_2 using deletion, insertions and substitutions.

$\Downarrow$ About Balanced Sequence of Parenthesis

Dyck Edit Distance Problem
Input: A string S over a set of parenthesis
Output: Closest S' to S w.r.t. edit distance such that S' is balanced

$$S = \textbf{((\{)]} \quad \longrightarrow \quad S' = \textbf{(\{ \})}$$

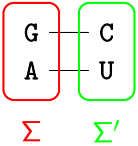
distance $= 3$

RNA Folding

RNA Folding
Input: A string S over alphabets $\Sigma \cup \Sigma'$, where each $\sigma \in \Sigma$ has a matching symbol $\sigma' \in \Sigma'$.
Output: Max. number of intersection-free connections between matching symbols σ, σ'.

RNA Folding

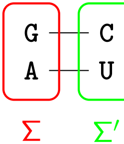
RNA Folding
Input: A string S over alphabets $\Sigma \cup \Sigma'$, where each $\sigma \in \Sigma$ has a matching symbol $\sigma' \in \Sigma'$.
Output: Max. number of intersection-free connections between matching symbols σ, σ'.

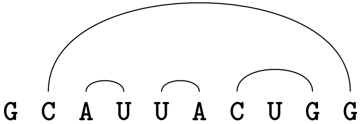


G C A U U A C U G G

RNA Folding

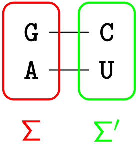
RNA Folding
Input: A string S over alphabets $\Sigma \cup \Sigma'$, where each $\sigma \in \Sigma$ has a matching symbol $\sigma' \in \Sigma'$.
Output: Max. number of intersection-free connections between matching symbols σ, σ'.

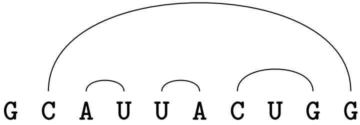




RNA Folding

RNA Folding
Input: A string S over alphabets $\Sigma \cup \Sigma'$, where each $\sigma \in \Sigma$ has a matching symbol $\sigma' \in \Sigma'$.
Output: Max. number of intersection-free connections between matching symbols σ, σ'.





Relation with edit distance: we can embed edit distance in RNA folding.

Algorithms for Dyck Edit Distance and RNA Folding

References	Time	Remarks
AP72, NJ80	$\mathcal{O}(n^3)$	
VGF14	$\mathcal{O}(n^3/\log n)$	
BGSVW17	$\mathcal{O}(n^{2.8603})$	
CDXW22	$\mathcal{O}(n^{2.687})$	
ABVW18	$\Omega(n^{3-o(1)})$	under k-Clique Detection, combinatorial

Algorithms for Dyck Edit Distance and RNA Folding

References	Time	Remarks
AP72, NJ80	$\mathcal{O}(n^3)$	
VGF14	$\mathcal{O}(n^3 / \log n)$	
BGSVW17	$\mathcal{O}(n^{2.8603})$	
CDXW22	$\mathcal{O}(n^{2.687})$	
ABVW18	$\Omega(n^{3-o(1)})$	under k-Clique Detection, combinatorial

Question 3: How about *Dynamic* Dyck Edit Distance and RNA Folding?

Our Results: Bad News...

Our Results

Question 1: How about *Dynamic* Tree Edit Distance?

Our Results

Question 1: How about *Dynamic* Tree Edit Distance?

Theorem

For any $\varepsilon > 0$, there are no dynamic algorithms:

- for weighted TED with $\mathcal{O}(n^{3-\varepsilon})$ update time, unless Weighted 4-Clique Conj. fails;
- for unweighted TED with $\mathcal{O}(n^{2-\varepsilon})$ (comb.) updated time, unless k-Clique Detection Conj. fails.

Our Results

Question 1: How about *Dynamic* Tree Edit Distance?

Theorem

For any $\varepsilon > 0$, there are no dynamic algorithms:

- for weighted TED with $\mathcal{O}(n^{3-\varepsilon})$ update time, unless Weighted 4-Clique Conj. fails;
- for unweighted TED with $\mathcal{O}(n^{2-\varepsilon})$ (comb.) updated time, unless k-Clique Detection Conj. fails.

Question 2: Is Unweighted Edit Distance on trees harder than on strings?

Dynamic Unweighted Edit Distance $\neq$ Dynamic Unweighted Tree Edit Distance

Our Results

Question 1: How about *Dynamic* Tree Edit Distance?

Theorem

For any $\varepsilon > 0$, there are no dynamic algorithms:

- for weighted TED with $\mathcal{O}(n^{3-\varepsilon})$ update time, unless Weighted 4-Clique Conj. fails;
- for unweighted TED with $\mathcal{O}(n^{2-\varepsilon})$ (comb.) updated time, unless k-Clique Detection Conj. fails.

Question 2: Is Unweighted Edit Distance on trees harder than on strings?

Dynamic Unweighted Edit Distance $\neq$ Dynamic Unweighted Tree Edit Distance

Question 3: How about *Dynamic* Dyck Edit Distance and RNA Folding?

Our Results

Question 1: How about *Dynamic* Tree Edit Distance?

Theorem

For any $\varepsilon > 0$, there are no dynamic algorithms:

- for weighted TED with $\mathcal{O}(n^{3-\varepsilon})$ update time, unless Weighted 4-Clique Conj. fails;
- for unweighted TED with $\mathcal{O}(n^{2-\varepsilon})$ (comb.) updated time, unless k-Clique Detection Conj. fails.

Question 2: Is Unweighted Edit Distance on trees harder than on strings?

Dynamic Unweighted Edit Distance $\neq$ Dynamic Unweighted Tree Edit Distance

Question 3: How about *Dynamic* Dyck Edit Distance and RNA Folding?

Theorem

For any $\varepsilon > 0$, there are no dynamic algorithms for Dyck Edit Distance and RNA Folding with $\mathcal{O}(n^{3-\varepsilon})$ (comb.) updated time, unless k-Clique Detection Conj. fails

Thanks!