An Algorithm for Comparing Similarity Between Two Trees: Edit Distance with Gaps

Hangjun Xu
Department of Mathematics and Computer Science
Duke University

April 7, 2014
M.S. Defense Presentation

Shape Matching

- *Given two shapes, how to quantify the similarity between them*? For instance:
  - (0-dim) Point sets (e.g. data feature points in data mining);
  - (1-dim) Curves (e.g. trajectories);
  - (2-dim) Surfaces (e.g. faces, images).
- **clustering**.
- **classification**.

Edit Distance and String Matching/Sequence Alignment

- **Edit distance**, or **Levenshtein distance**, was first used to quantify similarity between two sequences of characters, also called "strings".
- One string is changed to another via a sequence of **edit operations** include: insert or delete a character, or substitute one character with another.
- Each edit operation has a cost, and the edit distance between two strings is defined to be the minimum cost of transforming one string to another.
- The edit distance gives rise to an **alignment** between two strings, i.e. a mapping between the characters in each string. A deleted character is aligned with a **blank** character. For instance,

s a - v - - e

s a l v a g e

Applications of String Matching

- Computational biology: DNA sequence comparison (strings of characters A, G, C, T).
- Natural language processing: spell checking that compares the words entered and the words in a dictionary.
- String searching: finding a query string in a string database.

Computation of String Edit Distance

- Strings S_1 and S_2 with m and n characters, respectively.
- Gap in an alignment: a longest consecutive blank characters.
- Gap penalty function.
- Linear gap penalty (each blank character in a gap is charged equally): $O(mn)$, using dynamic programming;
- Convex gap penalty (the first blank character in a gap is charged more than the others): $O(mn)$, using dynamic programming;
- Arbitrary gap penalty: $O(mn^2 + m^2n)$, using dynamic programming;
- Can be improved using parallel algorithms, etc.

A Related Problem: Trajectory Similarity

- A smooth (resp. continuous, C^1, etc) trajectory is a smooth (resp. continuous, C^1, etc) curve $\gamma : [0, 1] \longrightarrow \mathbb{R}^2$. One can sample points from a trajectory, and define the similarity between two trajectories to be the similarity between the two sequences of sampled points in $\mathbb{R}^2$.
- S. Sankararaman, P. Agarwal and T. Molhave defined a comparison model that combines string alignment and dynamic time warping.
- Sequence Alignment: noise-robust (gaps) but bad with non-uniform sample rate.
- Dynamic Time Warping (average Fréchet distance): Good with non-uniform sample rate (multiple points can be matched to a single point), but bad with noise.

Tree Comparison: Motivation

- Many (complicated) shapes have underlying tree structures (e.g. via deformation retract) that preserve some key topological and geometric properties (e.g. connectedness, genus, homotopy type, critical points, etc).
- Thus, we can reduce the comparison of such shapes to the comparison of their underlying tree structures (linear hence might be simpler) if the information we care about are preserved.
- The underlying tree structure of a shape can be thought of as its **skeleton**.

Example 1: Medial Axis

Let S be a region in $\mathbb{R}^2$ such that $C := \partial S$ is a planer curve. The medial axis consists of all **centers** of disks (w.r.t. Euclidean norm) contained in S that intersect C tangentially at least twice:

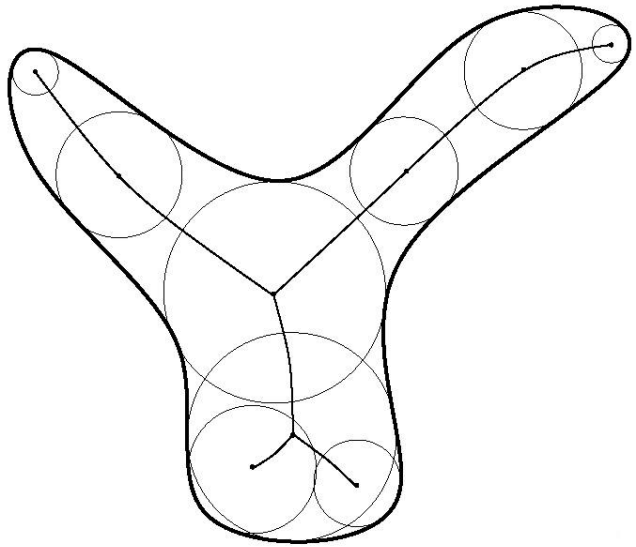


Figure: Picture from `http://www.lems.brown.edu/vision/Presentations/Wolter/figs.html`.

- Tree structure if C is piecewise polygonal.
- Topological skeleton: deformation retract.
- Robotics: effective motion planning.

Example 2: Contour Trees (revisited later)

Terrain and level sets:

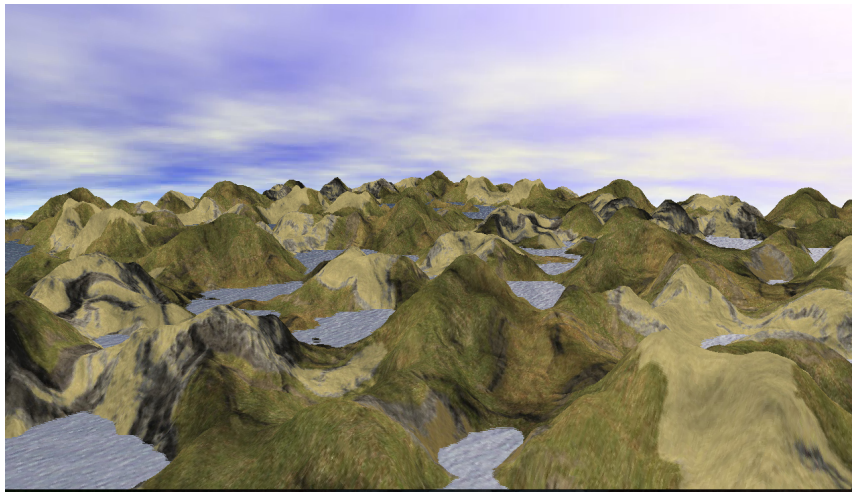


Figure: Picture by Tingran Gao and Hangjun Xu.

- A connected component of a level set is called a **contour**.
- Change of topology occur at **critical points** of the height function: local maximum (die), local minimum (born) and saddles (split/join).
- **Contour tree**: evolution of contours. **Nodes**: critical points; **Edges**: (u, v) if a contour appears at v and disappears at u.
- Terrain comparison $\longrightarrow$ contour tree comparison.

Classic Tree Edit Distance: Setups

- Tree T: **ordered** (sibling order is defined), **labeled** (each node has assigned a symbol from a finite alphabet Σ).
- **Edit Operations**:
  - (a) Rename. To rename one node label to another.
  - (b) Delete. To delete a node u, and all children of u become children of the parent of u, while maintaining the order.
  - (c) Insert. To insert a node u as a child of u'. A consecutive sequence of children of u' now becomes the children of u.

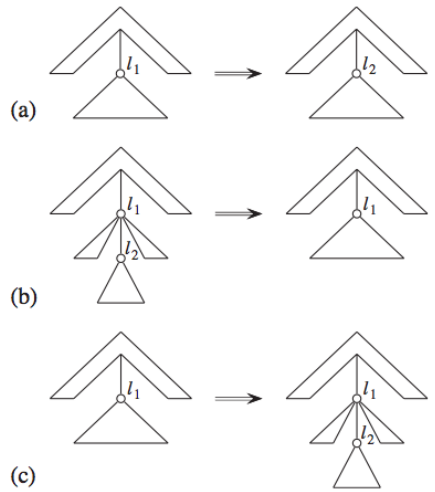


Figure: Picture from Philip Bille: *A Survey on Tree Edit Distance and Related Problems.*

Classic Tree Edit Distance: Setups Continued

Fix two ordered labeled trees T_1 and T_2 with vertex sets V_1 and V_2, respectively.

- A cost function is a metric $p : \Sigma \cup \{\perp\} \times \Sigma \cup \{\perp\} \longrightarrow \mathbb{R}^{\geq 0}$.
- $p(u, v)$ is the cost of relabeling u to v, $u \in V_1, v \in V_2$.
- $p(*, \perp)$ or $p(\perp, *)$ is the cost of deletion or insertion, respectively, where $\perp$ is a special character outside of Σ.
- $S = \{s_1, s_2, \cdots, s_n\}$: edit script taking T_1 to T_2. Cost of S is $C(S) := \sum_{i=1}^{n} C(s_i)$.

Definition

The edit distance between T_1 and T_2 is defined to be:
$\gamma(T_1, T_2) := \min\{C(S) | S$ *is an edit script taking* T_1 *to* $T_2\}$.

Classic Tree Edit Distance: Edit Script Mapping

A edit script gives rise to a mapping between two trees: graphical representation.

Definition (Mapping Between Two Trees)

$M \subset V_1 \times V_2$. For any (u, v) and (u', v') in M:

(1) *$u = u'$ if and only if $v = v'$;*

(2) *u is to the left of u' if and only if v is to the left of v';*

(3) *u is an ancestor of u' if and only if v is an ancestor of v'.*

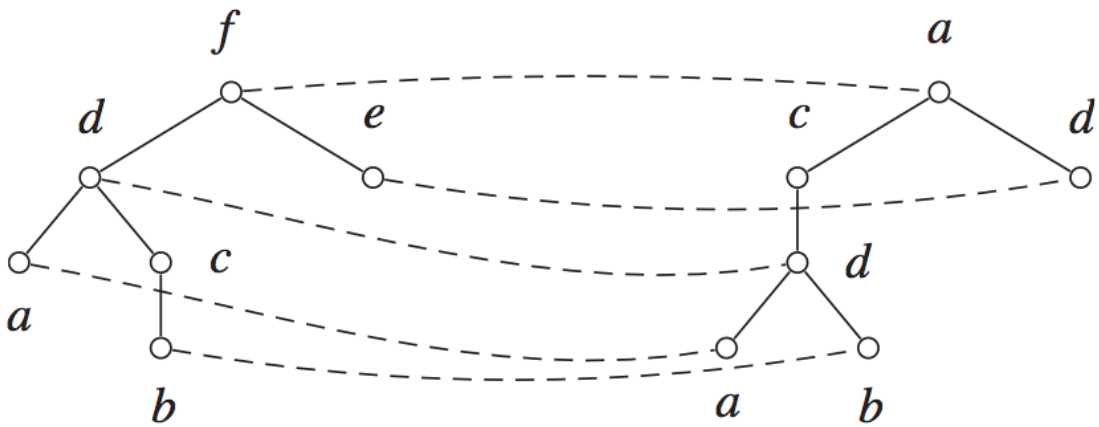


Figure: Picture from Philip Bille: *A Survey on Tree Edit Distance and Related Problems.*

Classic Tree Edit Distance Algorithms: an overview

Let $m := |T_1|$, $n := |T_2|$, $D_i := depth(T_i)$ and $L_i := |leaves(T_i)|$, $i = 1, 2$.

- Straightforward dynamic programming algorithm: $O(m^2n^2)$;
- Zhang and Shasha, 1989: $O(mn \min\{D_1, L_1\} \min\{D_2, L_2\})$, using **key roots**;
- Klein, 1998: $O(m^2 n \log n)$, using **path decomposition**;
- Chen, 2001: $O(mn + L_1^2 n + L_1^{2.5} L_2)$;
- Dulucq, Touzet, 2003: $O(mn \log^2 n)$;
- Demaine et al, 2009: $O(m^2 n)$.

Classic Tree Edit Distance Simple DP Algorithm Part I: Terminology

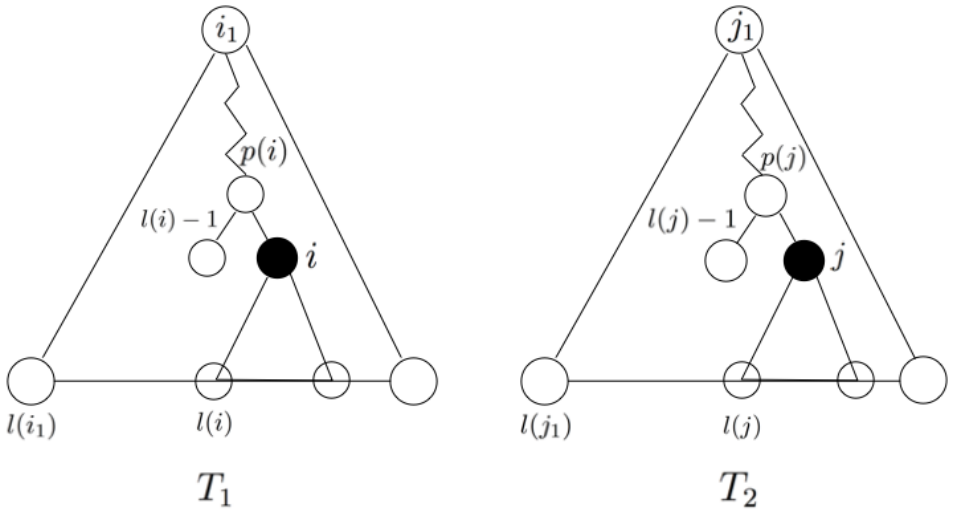
- Fix left-to-right **postorder** traversal $\longrightarrow$ numbering among all the nodes.
- $T[i]$: the i^{th} node.
- $l(i)$: the index of the leftmost leaf descendant of the subtree rooted at $T[i]$. (e.g. $T[i]$ is a leaf $\Longrightarrow l(i) = i$).
- $r(i)$: the index of the rightmost leaf descendant of the subtree rooted at $T[i]$. (e.g. $T[i]$ is a leaf $\Longrightarrow r(i) = i$).
- $p(i)$: parent of $T[i]$; and $desc(i)$: descendants of $T[i]$.
- $Q[i..i', j..j']$: edit distance between $T_1[i..i']$ and $T_2[j..j']$.

Classic Tree Edit Distance Simple DP Algorithm Part II: Recurrence

Theorem

For any $i \in desc(i_1)$ *and* $j \in desc(j_1)$,

$$Q[l(i_1)..i, l(j_1)..j] = \min \begin{cases} Q[l(i_1)..i-1, l(j_1)..j] + p(i, \perp) \\ Q[l(i_1)..i, l(j_1)..j-1] + p(\perp, j) \\ Q[l(i_1)..l(i)-1, l(j_1)..l(j)-1] \\ +Q[l(i)..i-1, l(j)..j-1] + p(i, j) \end{cases}$$



Edit Distance with Gaps: Motivation

- Recall gaps in string matching: longest consecutive blank characters. Analogues for trees?
- Noise in the input can be modeled by gaps.

Two questions:

1. What is a "good" gap model?
2. What is a "good" gap penalty function?

One heuristic for gap penalty function: *a single large gap is more probable than isolated smaller gaps*.

- A function $w : \mathbb{Z}^+ \longrightarrow \mathbb{R}^{\geq 0}$ is convex if $w(k_1 + k_2) \leq w(k_1) + w(k_2)$, $k_1, k_2 \in \mathbb{Z}^+$.
- $w(k) := a + bk$ is convex if $a \geq 0, b > 0$.
- Classic edit distance: linear gap penalty.

Complete Subtree Gap Edit Distance Part I: Setups

Definition (Complete Subtree Gap Model, Touzet, 2003)

Given a tree T with vertex set V. A gap g_v of T is the complete subtree rooted at some vertex $v \in V$.

- Admissible Edit Operations:
  1. Relabel a node;
  2. Delete a gap;
  3. Insert a gap.
- We will use **affine** gap penalty function.
- Objective function:

$$\gamma(M) := \sum_{(u,v)\in M} p(u,v) + \sum_{g\in G} a + b|g|, \quad u \in V_1 \; v \in V_2, \; (0.1)$$

where $M : T_1 \longrightarrow T_2$, G is the set of all gaps in M.

Complete Subtree Gap Edit Distance Part II: Auxiliary Functions

- If u is a gap node, then all its descendants are gap nodes.
- Starting a gap and continuing a preexisting gap have different penalty.
- How to determine if a gap node is starting or continuing a gap? **Preorder traversal**!

Definition

For $1 \leq i' \leq i \leq m := |T_1|$, *and* $1 \leq j' \leq j \leq n := |T_2|$, *set:*

$$\begin{cases} Q[i'..i, j'..j] := \gamma(T_1[i'..i], T_2[j'..j]); \\ Q_{\perp *}[i'..i, j'..j] := \gamma(T_1[i'..i], T_2[j'..j]) \text{ such that } i \to \perp; \\ Q_{* \perp}[i'..i, j'..j] := \gamma(T_1[i'..i], T_2[j'..j]) \text{ such that } \perp \to j \end{cases}$$

- Goal: compute $Q[1..m, 1..n]$.

Complete Subtree Gap Edit Distance Part III: Boundary Conditions

- $\emptyset$ = empty tree.
- Preorder traversal.
- Boundary conditions:

$$
\begin{aligned}
Q[\emptyset, \emptyset] &= 0 \\
Q[1..i, \emptyset] &= \infty, && \text{(for } 1 \leq i \leq m\text{)} \\
Q[\emptyset, 1..j] &= \infty, && \text{(for } 1 \leq j \leq n\text{)} \\
\\
Q_{\perp *}[1..i, \emptyset] &= a + bi, && \text{(for } 1 \leq i \leq m\text{)} \\
Q_{\perp *}[\emptyset, 1..j] &= \infty, && \text{(for } 1 \leq j \leq n\text{)} \\
\\
Q_{* \perp}[1..i, \emptyset] &= \infty, && \text{(for } 1 \leq i \leq m\text{)} \\
Q_{* \perp}[\emptyset, 1..j] &= a + bj, && \text{(for } 1 \leq j \leq n\text{)}
\end{aligned}
$$

Complete Subtree Gap Edit Distance Part IV: Recurrence

Theorem 3.1 (Recurrence of Auxiliary Matrices in Complete Subtree Gap Model). *Given the preorder ordering on the nodes of two ordered labeled trees T_1 and T_2. Fix nodes $i_1 \in V_1, j_1 \in V_2$. For any $i \in desc(i_1)$ and $j \in desc(j_1)$, we have the following recurrence relations:*

$$Q[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i-1, j_1..j-1] + p(i,j) \\ Q_{\perp *}[l_1..i, j_1..j] \\ Q_{* \perp}[i_1..i, j_1..j] \end{cases} \tag{3.3.3}$$

$$Q_{\perp *}[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i-1, j_1..j] + (a+b) \\ Q_{\perp *}[i_1..p(i), j_1..j] + b(i-p(i)) \end{cases} \tag{3.3.4}$$

$$Q_{* \perp}[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i, j_1..j-1] + (a+b) \\ Q_{* \perp}[i_1..i, j_1..p(j)] + b(j-p(j)) \end{cases} \tag{3.3.5}$$

Here $p(i)$ (resp. $p(j)$) is the index of parent node (if exists) of i (resp. j).

Complete Subtree Gap Edit Distance Part IV: Recurrence Continued

Illustration of recurrence for $Q_{\perp *}$, case II.

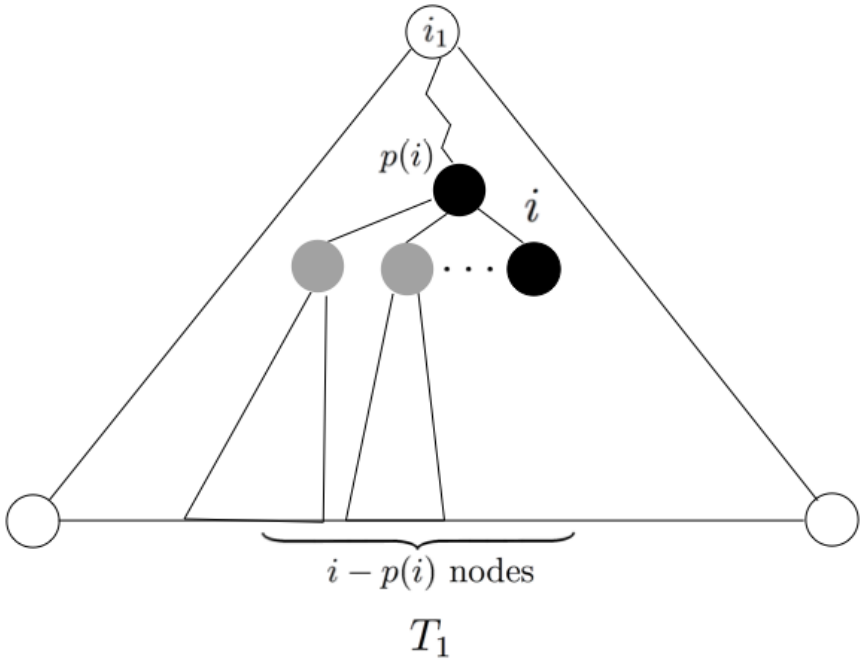


Figure: $p(i)$ and i are both gap nodes.

$$Q_{\perp *}[i_1..i, j_1..j] = Q_{\perp *}[i_1..p(i), j_1..j] + b(i - p(i)).$$

Complete Subtree Gap Edit Distance Part V: Algorithm

- Let $\text{treedist}(i_1, j_1) := Q[i_1..r(i_1), j_1..r(j_1)]$.

Algorithm 2 Complete Subtree Gap Tree Edit Distance

Input: Tree T_1 and T_2
Output: $\text{treedist}(i_1, j_1)$, where $1 \leqslant i_1 \leqslant m := |T_1|$, and $1 \leqslant j_1 \leqslant n := |T_2|$
Preprocessing: Compute the index of the parent of each node and the r function
for $(i_1, j_1 = 1; i_1 \leqslant m, j_1 \leqslant n; i_1{+}{+}, j_1{+}{+})$ **do**
  for $(i = i_1; i \leqslant r(i_1); i{+}{+})$ **do**
    for $(j = j_1; j \leqslant r(j_1); j{+}{+})$ **do**
      Compute $\text{treedist}(i_1, j_1)$ by first compute $Q_{\perp *}[i_1..i, j_1..j]$,
      then compute $Q_{* \perp}[i_1..i, j_1..j]$
    end for
  end for
end for

- Running time $O(m^2 n^2)$.

General Gap Edit Distance of Binary Trees Part I: Setups

Definition (General Gaps Model, Touzet, 2003)

Given an ordered labeled tree T with vertex set V and edge set E. A gap g is a tree with vertex set a subset of V and edges in E whose both end points lie in that subset. That is, g is a subtree of T.

- Admissible Edit Operations:
  1. Relabel a node;
  2. Delete a gap: descendants of a gap will become children of the parent of the root of the gap;
  3. Insert a gap.
- Touzet, 2003: Computation of general gap edit distance is NP-hard even for affine gap penalty.
- Restrict to binary trees. Same auxiliary functions and boundary conditions.

General Gap Edit Distance of Binary Trees Part II: Recurrence

Theorem 3.2 (Recurrence of Auxiliary Matrices in General Gap Model for Binary Trees). *Given the preorder ordering on the nodes of two ordered labeled trees T_1 and T_2. Fix nodes $i_1 \in V_1, j_1 \in V_2$. For any $i \in desc(i_1)$ and $j \in desc(j_1)$, we have the following recurrence relations:*

$$Q[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i-1, j_1..j-1] + p(i,j) \\ Q_{\perp *}[i_1..i, j_1..j] \\ Q_{* \perp}[i_1..i, j_1..j] \end{cases} \tag{3.4.5}$$

$$Q_{\perp *}[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i-1, j_1..j] + (a+b) \\ Q_{\perp *}[i_1..i-1, j_1..j] + b \\ \min\limits_{j_1 \leqslant k \leqslant j} \{Q_{\perp *}[i_1..p(i), j_1..k] \\ \qquad + Q[p(i)+1..i-1, k+1..j] + b\} \end{cases} \tag{3.4.6}$$

$$Q_{* \perp}[i_1..i, j_1..j] = \min \begin{cases} Q[i_1..i, j_1..j-1] + (a+b) \\ Q_{* \perp}[i_1..i, j_1..j-1] + b \\ \min\limits_{i_1 \leqslant k \leqslant i} \{Q_{* \perp}[i_1..k, j_1..p(j)] \\ \qquad + Q(k+1..i, p(j)+1..j-1) + b\} \end{cases} \tag{3.4.7}$$

Here $p(i)$ (resp. $p(j)$) is the index of parent node (if exists) of i (resp. j).

General Gap Edit Distance of Binary Trees Part III: Proof of Recurrence for $Q_{\perp *}$

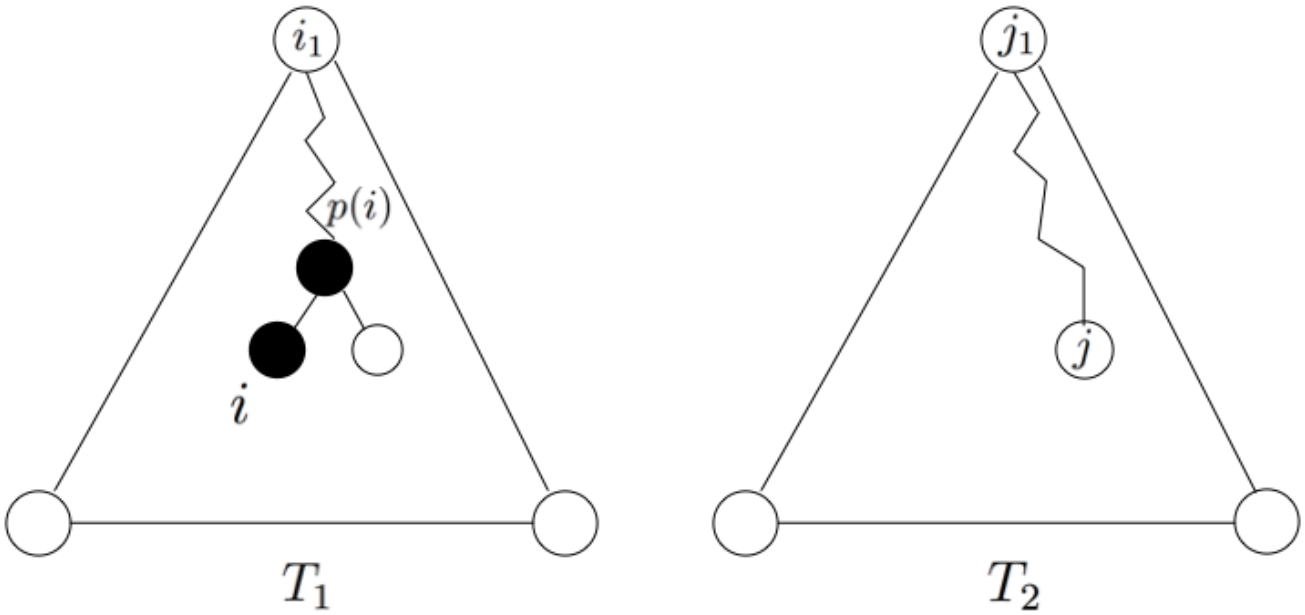


Figure: $p(i)$ is a gap node and i is its left child. Gap nodes are labeled black.

$$Q_{\perp *}[i_1..i, j_1..j] = Q_{\perp *}[i_1..i-1, j_1..j] + b.$$

General Gap Edit Distance of Binary Trees Part III: Proof of Recurrence for $Q_{\perp *}$ Continued

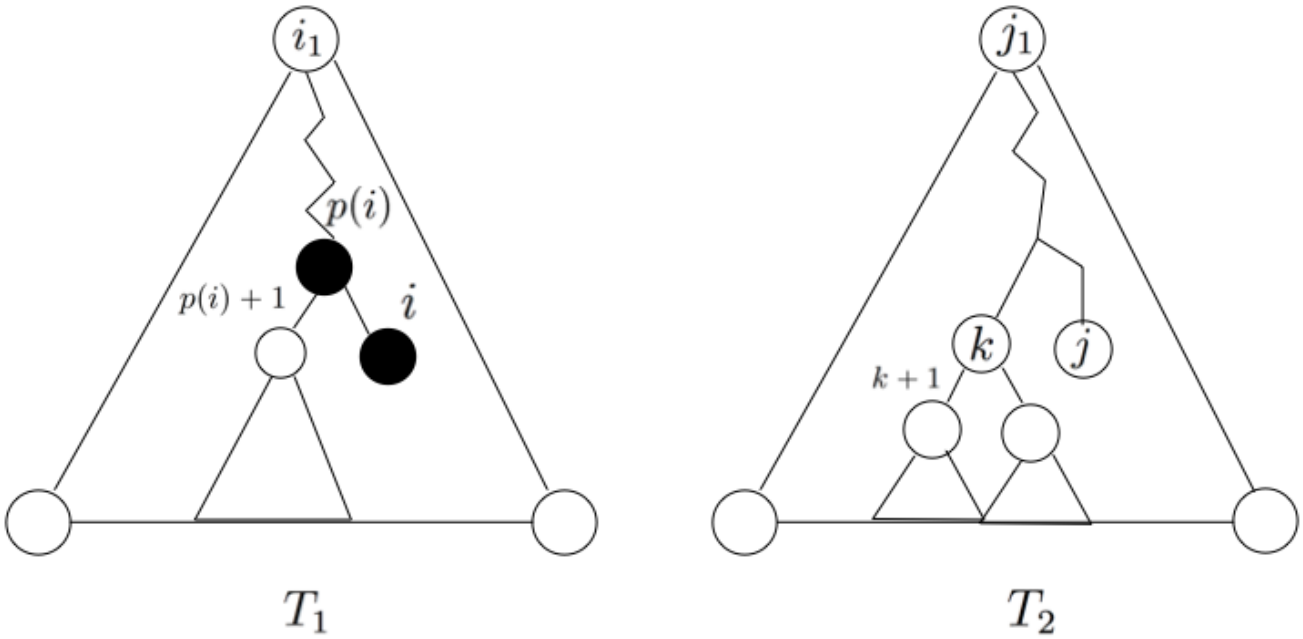


Figure: $p(i)$ is a gap node and i is its right child. Gap nodes are labeled black.

$$Q_{\perp *}[i_1..i, j_1..j] = Q_{\perp *}[i_1..p(i), j_1..k] + Q[p(i)+1..i-1, k+1..j] + b.$$

General Gap Edit Distance of Binary Trees Part IV: Algorithm

- Let treedist$(i_1, j_1) := Q[i_1..r(i_1), j_1..r(j_1)]$.

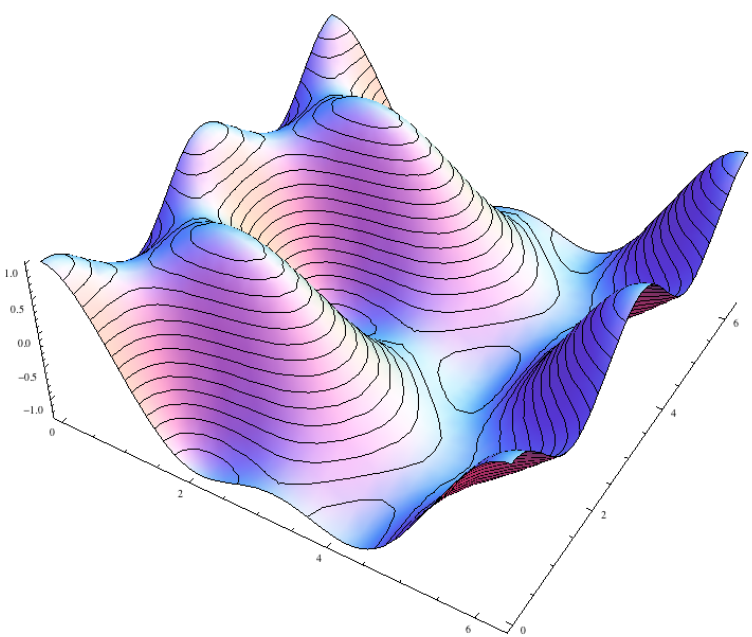
Algorithm 3 General Gap Tree Edit Distance

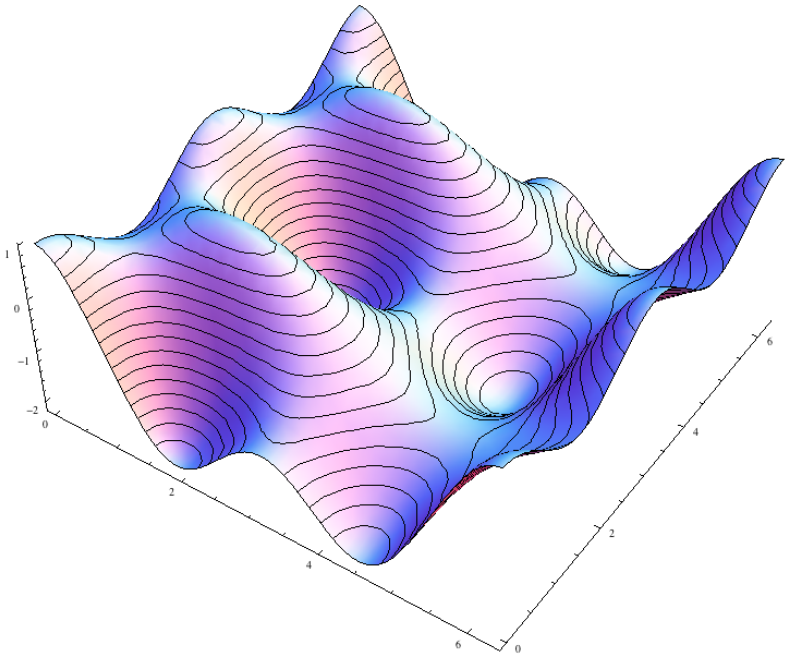
Input: Tree T_1 and T_2
Output: treedist(i_1, j_1), where $1 \leqslant i_1 \leqslant m := |T_1|$, and $1 \leqslant j_1 \leqslant n := |T_2|$
Preprocessing: Compute the index of the parent of each node and the r function
for $(i_1, j_1 = 1; i_1 \leqslant m, j_1 \leqslant n; i_1{+}{+}, j_1{+}{+})$ **do**
  for $(i = i_1; i \leqslant r(i_1); i{+}{+})$ **do**
    for $(j = j_1; j \leqslant r(j_1); j{+}{+})$ **do**
      Compute treedist(i_1, j_1) by first compute $Q_{\perp *}[i_1..i, j_1..j]$,
      then compute $Q_{*\perp}[i_1..i, j_1..j]$
    end for
  end for
end for

- Running time $O(m^3n^2 + m^2n^3)$.

Applications of Complete Subtree Edit Distance to Contour Tree Comparison: Work in Progress

Questions: How do we compare two terrains?





Applications of Complete Subtree Edit Distance to Contour Tree Comparison: Work in Progress Continued

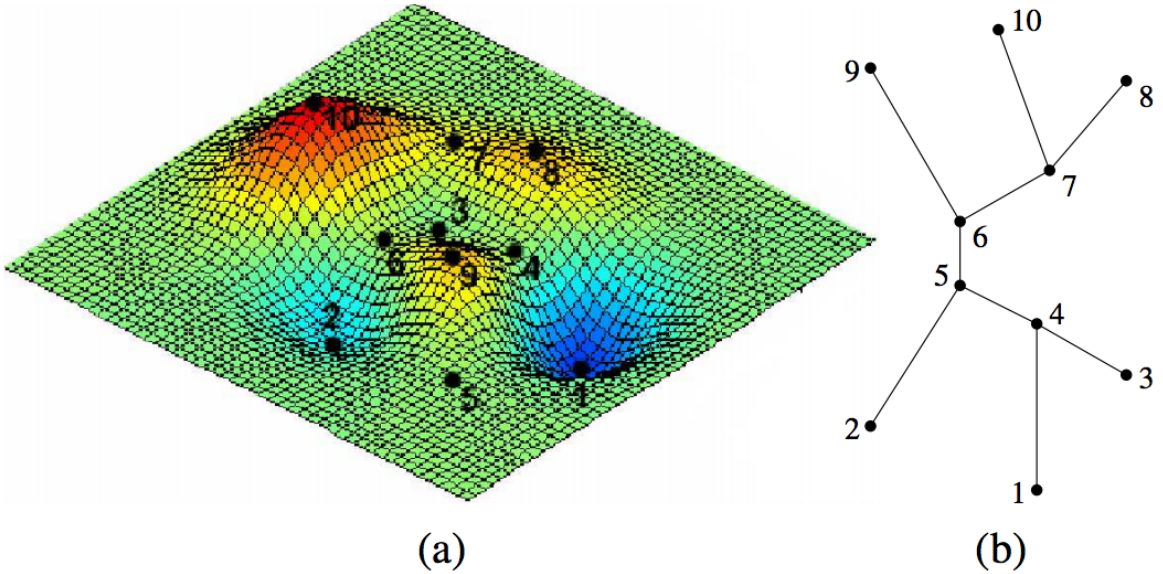


Figure: Terrain with its contour tree. Picture taken from P. Agarwal et al: *I/O-Effcient Batched Union-Find and Its Applications to Terrain Analysis*

- Noise in the input correspond to complete subtrees of the contour tree.
- Gap penalty: persistence, height, volume.

Open Problems and Future Works

- General gap edit distance for trees with fixed degree.
- Constraints of gap sizes.
- Geometric comparisons for trees.