Analyzing Edit Distance on Trees

Tree Swap Distance is Intractable

Martin Berglund, mbe@cs.umu.se

Department of Computing Science
Natural and Formal Languages (NFL) Group

August 29, 2011

Outline

- Recall string correction problem (Damerau-Levenshtein)
- Recall tree correction problem (Selkow)
- Define a *swap* operation for trees and discuss the problem of integrating it in Selkow tree correction problems
- Show that swaps in tree correction is intractable in general through a three step reduction

Background: String edit distance

- Edit distance on strings is well known. The operations
  - Delete a single symbol anywhere in a string (abc $\Rightarrow$ ac)
  - Insert a single symbol anywhere in a string (abc $\Rightarrow$ adbc)
  - (Replace a single symbol by another: ignored here)

  make up Levenshtein distance
- Damerau-Levenshtein distance adds a swap operation (abc $\Rightarrow$ bac or abc $\Rightarrow$ acb)
- The distance from $s \in \Sigma^*$ to $s' \in \Sigma^*$ is the number of operations necessary to transform s into s', the decision problem becomes:

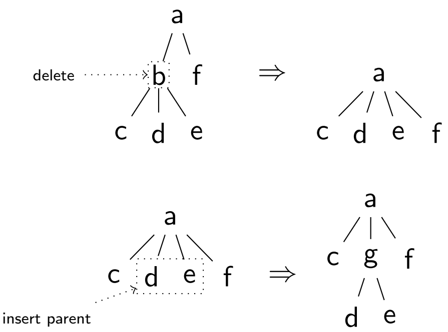
Damerau-Levenshtein String Correction Problem

Given $s, s' \in \Sigma^*$ and $k \in \mathbb{N}$, can s be turned into s' by performing at most k symbol *deletions*, *insertions*, and *swaps*?

Tree correction was defined by Selkow in '77:

Tree Correction Problem

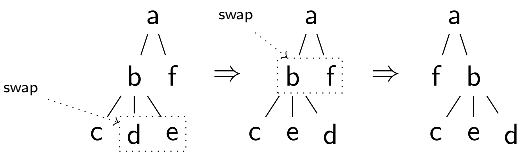
Given two trees t and t' and $k \in \mathbb{N}$, can t be turned into t' by performing at most k node *deletions*, and *insertions*?



Efficient algorithms available (Zhang-Shasha for example)

Adding swaps to tree correction?

- Selkow tree correction only has deletions and insertions
- Swaps in trees are easy to define though:



- Having swaps is also useful in all kinds of applications
- So why isn't it done? The correction problem becomes NP-complete!

No swaps in tree edit distance?

Unordered tree inclusion (NP-complete)

Given two *unordered* trees t and t', can t' be obtained from t by a sequence of deletions?

- We can reduce to this to tree correction as follows
- Set budget $k = (1 + |t| - |t'|)|t|^2 - 1$
- Replace each node in both t and t' by a unary tree of height $|t|^2$, simulating a cost of $|t|^2$ for deletions/insertions
- Then the budget allows at most $|t| - |t'|$ deletions/insertions, so no insertions possible
- The left-over budget $|t|^2 - 1$ is enough to make *any* reordering using swaps
- In summary, only deletes can be used and t can be freely reordered, so tree correction with swaps is NP-complete

So, what to do about tree swaps?

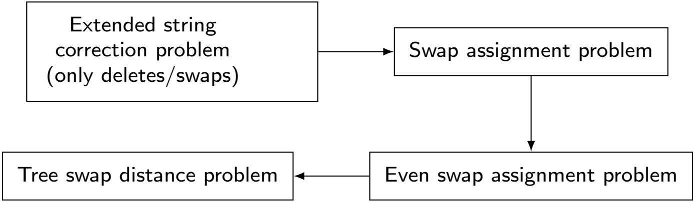
- What now? Subtree movements is desirable in real applications
- Polynomial algorithms exist which weaken the swap (each tree may only participate in a constant number of swaps: Barnard et al., '95)
- How about the other route, where swaps are allowed but the other operations are weakened?
- The simplest and most extreme approach: allowing *only* swaps is *also* NP-complete! Let's look at why

Tree swap distance: NP-complete

Tree swap distance problem

Given two trees t and t' and $k \in \mathbb{N}$, can t be turned into t' by performing at most k *swaps* on t?

We demonstrate NP-completeness with a sequence of reductions



The first is known to be strongly NP-complete, the rest are new

Delete/swap string correction and swap assignment

Wagner generalized the string correction problem where each operation has a cost. Cases where *inserts* has cost ∞ turns out strongly NP-complete:

Extended string correction problem, deletes/swaps only

Given $s, s' \in \Sigma^*$ and $k \in \mathbb{N}$ can s be transformed into s' by deleting symbols from s and then performing at most k swaps?

We reduce this to the intermediary problem:

Swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$ and $k \in \mathbb{N}$, is there a sequence of n swaps of adjacent rows in M such that $k \geq n + \sum diag(M)$?

Basically: swap rows to get a small diagonal

The delete/swap correction $\rightarrow$ swap assignment reduction

Take the delete/swap correction problem $s = aacb$, $s' = abc$, and $k = 1$, this constructs the swap assignment problem:

$$M = \left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right], k' = 5$$

$\underline{a}acb$

$$\left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right]$$

The delete/swap correction $\rightarrow$ swap assignment reduction

Take the delete/swap correction problem $s = aacb$, $s' = abc$, and $k = 1$, this constructs the swap assignment problem:

$$M = \left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right], k' = 5$$

$\underline{a}acb$

$$\left[\begin{array}{ccc|c} 0 & 6 & 6 & 2 \\ 0 & 6 & 6 & 1 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right]$$

The delete/swap correction $\rightarrow$ swap assignment reduction

Take the delete/swap correction problem $s = aacb$, $s' = abc$, and $k = 1$, this constructs the swap assignment problem:

$$M = \left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right], k' = 5$$

$\underline{a}acb$

$$\left[\begin{array}{ccc|c} 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 0 & 6 & 6 & 1 \\ 6 & 0 & 6 & 4 \end{array}\right]$$

The delete/swap correction $\rightarrow$ swap assignment reduction

Take the delete/swap correction problem $s = aacb$, $s' = abc$, and $k = 1$, this constructs the swap assignment problem:

$$M = \left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right], k' = 5$$

$a\underline{cb}$

$$\left[\begin{array}{ccc|c} 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \\ 0 & 6 & 6 & 1 \end{array}\right]$$

The delete/swap correction $\rightarrow$ swap assignment reduction

Take the delete/swap correction problem $s = aacb$, $s' = abc$, and $k = 1$, this constructs the swap assignment problem:

$$M = \left[\begin{array}{ccc|c} 0 & 6 & 6 & 1 \\ 0 & 6 & 6 & 2 \\ 6 & 6 & 0 & 3 \\ 6 & 0 & 6 & 4 \end{array}\right], k' = 5$$

abc

$$\left[\begin{array}{ccc|c} 0 & 6 & 6 & 2 \\ 6 & 0 & 6 & 4 \\ 6 & 6 & 0 & 3 \\ 0 & 6 & 6 & 1 \end{array}\right]$$

The general reduction shows swap assignment strongly NP-complete

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 2 & 3 & 3 \\ 9 & 4 & 12 \\ 1 & 2 & 8 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \left[\begin{array}{cccccc} 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 16 & 0 & 2 & 16 & 8 & 16 \\ \hline 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{array}\right], k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 2 & 3 & 3 \\ 1 & 2 & 8 \\ 9 & 4 & 12 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 0 & 2 & 16 & 8 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ \hline 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 2 & 3 & 3 \\ 9 & 4 & 12 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 16 & 0 & 2 & 16 & 8 & 16 \\ 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ \hline 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d\times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \left[\begin{array}{cccccc} 16 & 0 & 2 & 16 & 8 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 2 & 16 & 16 & 2 & 16 & 2 \\ \hline 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{array}\right], k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 16 & 0 & 2 & 16 & 8 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 0 & 0 & 16 & 16 & 16 & 16 \\ 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 16 & 0 & 2 & 16 & 8 & 16 \\ 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 16 & 0 & 2 & 16 & 8 & 16 \\ 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 2 & 16 & 16 & 2 & 16 & 2 \\ 16 & 16 & 16 & 16 & 0 & 0 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 16 & 0 & 2 & 16 & 8 & 16 \\ 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \\ 2 & 16 & 16 & 2 & 16 & 2 \end{bmatrix}, k' = 14$$

Swap assignment $\rightarrow$ even swap assignment reduction

A simple modification of swap assignment:

Even swap assignment problem

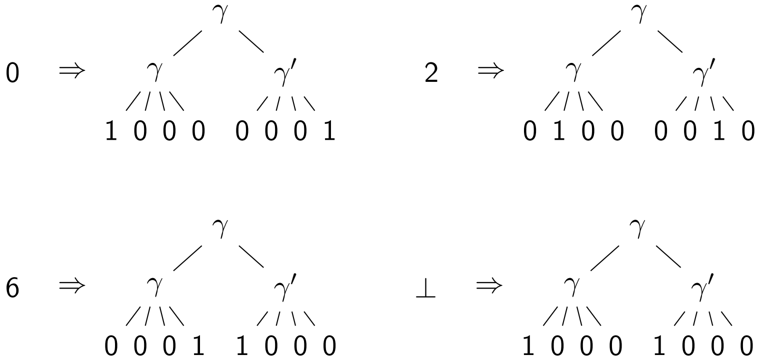
Given a square matrix $M \in \mathbb{N}_{d \times d}$, *containing only even numbers*, and $k \in \mathbb{N}$, can adjacent rows in M be swapped n times such that $k \geq n + \sum diag(M)$?

Reducing swap assignment to even swap assignment is done by rounding numbers down to be even and adding rows which simulate the odd costs:

$$\begin{bmatrix} 1 & 2 & 8 \\ 9 & 4 & 12 \\ 2 & 3 & 3 \end{bmatrix}, k = 11 \quad \Rightarrow \quad \begin{bmatrix} 0 & 0 & 16 & 16 & 16 & 16 \\ 16 & 0 & 2 & 16 & 8 & 16 \\ 16 & 8 & 4 & 16 & 12 & 16 \\ 16 & 16 & 0 & 0 & 16 & 16 \\ 16 & 16 & 16 & 16 & 0 & 0 \\ 2 & 16 & 16 & 2 & 16 & 2 \end{bmatrix}, k' = 14$$

Even swap assignment problem $\rightarrow$ tree swap distance

This reduction requires us to represent a number as a tree:



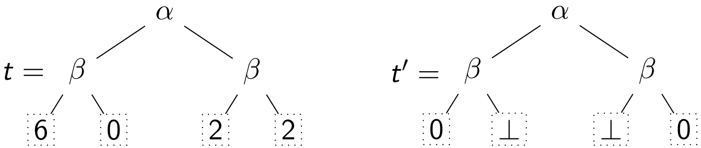
Notice how the swap distance between each is equal to the numerical difference, and $\perp$ is 3 swaps from all the others

Even swap assignment problem $\rightarrow$ tree swap distance

Take the even swap assignment problem

$$M = \begin{bmatrix} 6 & 0 \\ 2 & 2 \end{bmatrix}, k = 3.$$

M is translated into the tree t and t' is constructed



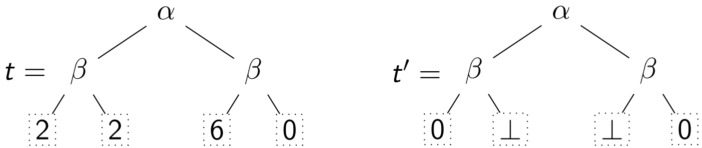
The constructed budget for the tree swap problem is $k' = 9$

Even swap assignment problem $\rightarrow$ tree swap distance

Take the even swap assignment problem

$$M = \begin{bmatrix} 2 & 2 \\ 6 & 0 \end{bmatrix}, k = 3.$$

M is translated into the tree t and t' is constructed



The constructed budget for the tree swap problem is $k' = 9$

With the one swap performed both problems are exactly solved

From the general reduction it follows that Tree swap distance problem is NP-complete

Summary

In summary we have seen:

- Tree edit distance, in the form of the tree correction problem, is both useful and well-known but only has deletion and insertion operators
- Adding subtree movement operators to these makes the correction problem intractable
- A correction problem using *only* swaps *also* turns out to be intractable in the case of trees
- This suggests that different subtree movement operations should be considered (linear distance?)
- The fact that tree swap distance is NP-complete may be helpful for analyzing other problems, since it is simple to define

In summary we have seen:

- Tree edit distance, in the form of the tree correction problem, is both useful and well-known but only has deletion and insertion operators
- Adding subtree movement operators to these makes the correction problem intractable
- A correction problem using *only* swaps *also* turns out to be intractable in the case of trees
- This suggests that different subtree movement operations should be considered (linear distance?)
- The fact that tree swap distance is NP-complete may be helpful for analyzing other problems, since it is simple to define

Thanks for listening