RTED

A Robust Algorithm for the Tree Edit Distance

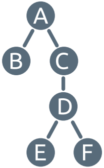
Mateusz Pawlik and Nikolaus Augsten

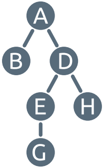
VLDB 2012, Istanbul, Turkey

Free University of Bozen-Bolzano

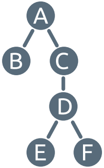
August 30, 2012

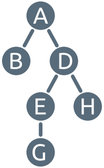
Tree edit distance





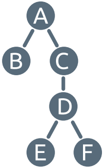
Tree edit distance

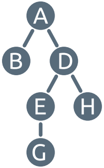




The tree edit distance is the minimum number of edit operations that transform one tree into another.

Tree edit distance



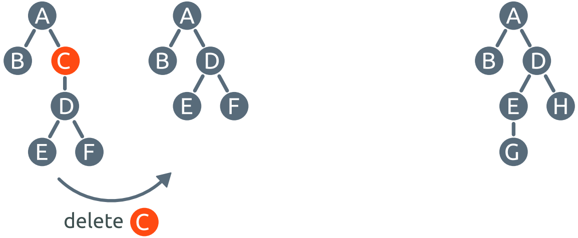


The tree edit distance is the minimum number of edit operations that transform one tree into another.

Three edit operations:

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Tree edit distance

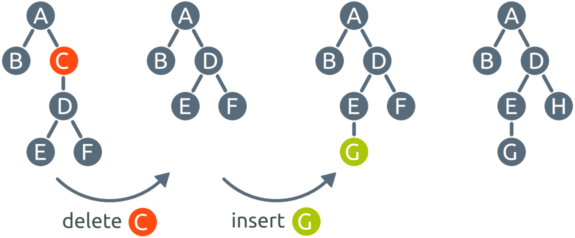


The tree edit distance is the minimum number of edit operations that transform one tree into another.

Three edit operations:

- delete a node

Tree edit distance

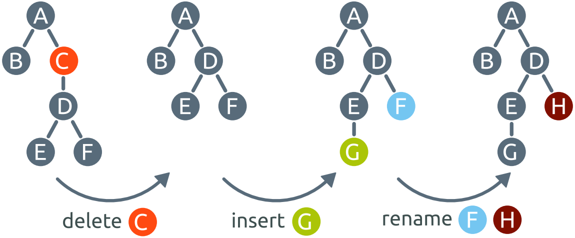


The tree edit distance is the minimum number of edit operations that transform one tree into another.

Three edit operations:

- delete a node
- insert a node

Tree edit distance

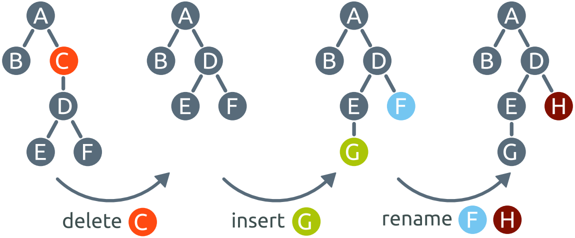


The tree edit distance is the minimum number of edit operations that transform one tree into another.

Three edit operations:

- delete a node
- insert a node
- rename the label of a node

Tree edit distance



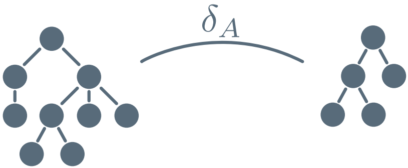
The tree edit distance is the minimum number of edit operations that transform one tree into another.

Three edit operations:

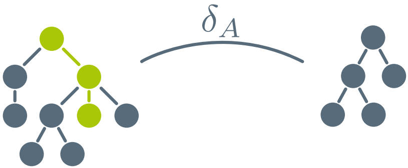
- delete a node
- insert a node
- rename the label of a node

Each operation has a cost.

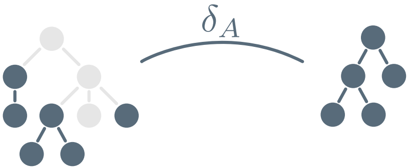
Cost of distance computation



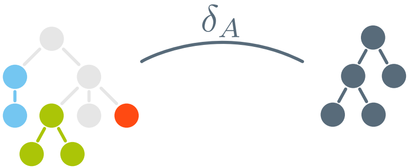
Cost of distance computation



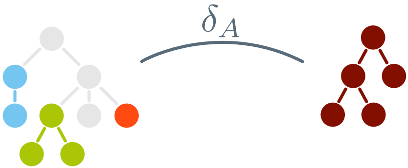
Cost of distance computation



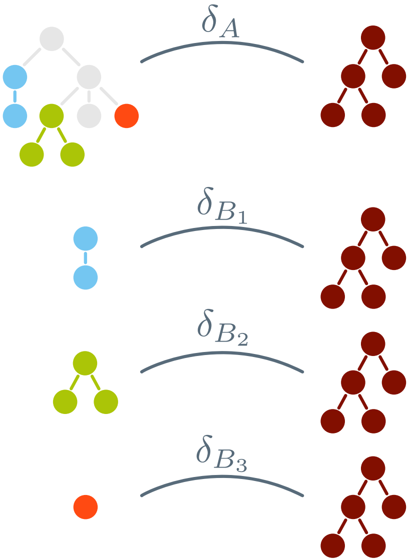
Cost of distance computation



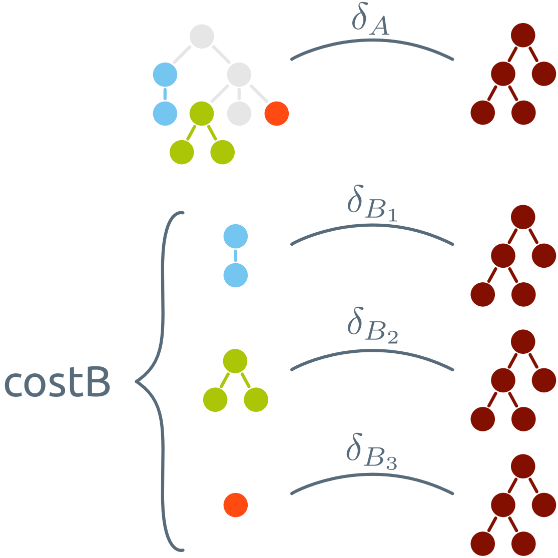
Cost of distance computation



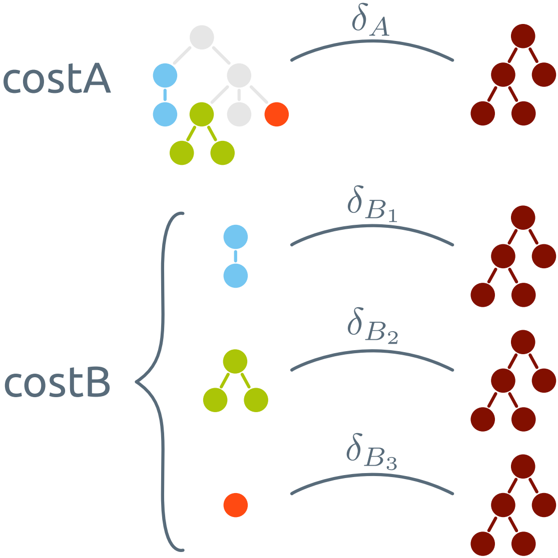
Cost of distance computation



Cost of distance computation



Cost of distance computation



Path strategies

Paths are used recursively to decompose trees into subproblems.

Path strategies

Paths are used recursively to decompose trees into subproblems.

Different paths result in different subproblems and different overall cost.

Path strategies

Paths are used recursively to decompose trees into subproblems.

Different paths result in different subproblems and different overall cost.

Path strategy: all paths used in a decomposition.

Path strategies

Paths are used recursively to decompose trees into subproblems.

Different paths result in different subproblems and different overall cost.

Path strategy: all paths used in a decomposition.

LRH strategy: allow only left, right and heavy paths.

Path strategies

Paths are used recursively to decompose trees into subproblems.

Different paths result in different subproblems and different overall cost.

Path strategy: all paths used in a decomposition.

LRH strategy: allow only left, right and heavy paths.

LRH strategies cover all the strategies used by the state-of-the-art algorithms.

State of the art

Fastest algorithms use LRH strategies.

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

- degenerate for some tree shapes

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

- degenerate for some tree shapes
- polynomial degree runtime difference

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

- degenerate for some tree shapes
- polynomial degree runtime difference

Zhang and Shasha (1989)

- always left path
- $O(n^4)$ time and $O(n^2)$ space
- efficient for balanced trees with $O(n^2 \log^2 n)$ time

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

- degenerate for some tree shapes
- polynomial degree runtime difference

Zhang and Shasha (1989)

- always left path
- $O(n^4)$ time and $O(n^2)$ space
- efficient for balanced trees with $O(n^2 \log^2 n)$ time

Demaine et al. (2007)

- heavy path in bigger tree
- $O(n^3)$ time and $O(n^2)$ space
- worst case is frequent for balanced trees

State of the art

Fastest algorithms use LRH strategies.

Use simple heuristics to determine the strategy.

Problem:

- degenerate for some tree shapes
- polynomial degree runtime difference

Zhang and Shasha (1989)

- always left path
- $O(n^4)$ time and $O(n^2)$ space
- efficient for balanced trees with $O(n^2 \log^2 n)$ time

Demaine et al. (2007)

- heavy path in bigger tree
- $O(n^3)$ time and $O(n^2)$ space
- worst case is frequent for balanced trees

State-of-the-art is not satisfactory.

Problem definition

Find a robust algorithm that combines the strengths of the state-of-the-art algorithms excluding their weaknesses.

FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN · BOLZANO

Problem definition

Find a robust algorithm that combines the strengths of the state-of-the-art algorithms excluding their weaknesses.

Requirements:

Problem definition

Find a robust algorithm that combines the strengths of the state-of-the-art algorithms excluding their weaknesses.

Requirements:

- space-efficient: $O(n^2)$ best known space complexity

Problem definition

Find a robust algorithm that combines the strengths of the state-of-the-art algorithms excluding their weaknesses.

Requirements:

- space-efficient: $O(n^2)$ best known space complexity
- optimal runtime: $O(n^3)$ optimal runtime among all possible strategies

Problem definition

Find a robust algorithm that combines the strengths of the state-of-the-art algorithms excluding their weaknesses.

Requirements:

- space-efficient: $O(n^2)$ best known space complexity
- optimal runtime: $O(n^3)$ optimal runtime among all possible strategies
- robust: always uses the best possible strategy for a given pair of trees

RTED - robust algorithm for the tree edit distance

RTED = optimal strategy + GTED

RTED - robust algorithm for the tree edit distance

RTED = optimal strategy + GTED

step1: compute optimal LRH strategy for given pair of trees

RTED - robust algorithm for the tree edit distance

RTED = optimal strategy + GTED

step1: compute optimal LRH strategy for given pair of trees

step2: feed optimal strategy into GTED

RTED - robust algorithm for the tree edit distance

RTED = optimal strategy + GTED

step1: compute optimal LRH strategy for given pair of trees

step2: feed optimal strategy into GTED

GTED:

- executes any LRH strategy in $O(n^2)$ space
- generalizes all state-of-the-art approaches

RTED - robust algorithm for the tree edit distance

RTED = optimal strategy + GTED

step1: compute optimal LRH strategy for given pair of trees

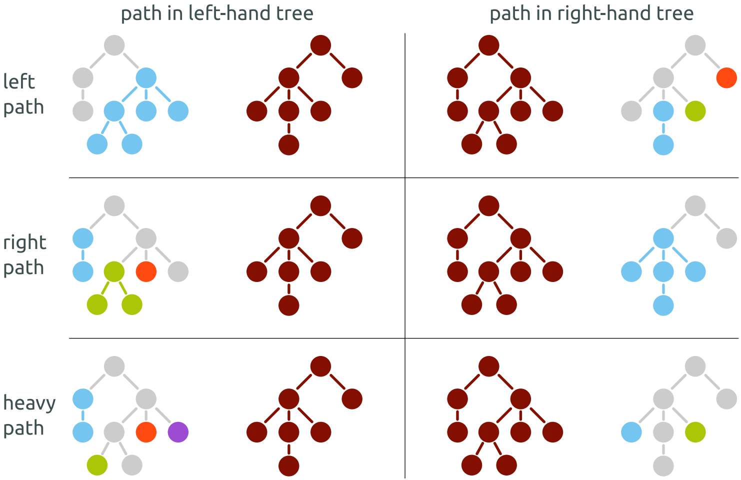
step2: feed optimal strategy into GTED

GTED:

- executes any LRH strategy in $O(n^2)$ space
- generalizes all state-of-the-art approaches

key problem - compute the optimal strategy efficiently

Optimal strategy - six choices



Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems
- pick the minimal cost

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems
- pick the minimal cost

$$\text{cost}(p) =$$

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems
- pick the minimal cost

$$\mathrm{cost}(p) = \qquad \mathrm{costA} + \mathrm{costB}$$

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems
- pick the minimal cost

$$\mathrm{cost}(p) = \min_{\text{six choices}} [\,\mathrm{costA} + \mathrm{costB}\,]$$

Optimal strategy - cost formula

The recursive cost formula performs a search in the space of all possible LRH strategies.

At each recursive step:

- six choices to get the subproblems
- pick the minimal cost

$$\text{cost}(p) = \min_{\text{six choices}} [\, \text{costA} + \underbrace{\text{costB}}_{\sum_{s \in subprob.} \text{cost}(s)} \,]$$

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

$O(n^3)$ time complexity

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

$O(n^3)$ time complexity

- cost of each problem computed once (quadratic number)

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

$O(n^3)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

$O(n^3)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB
- costA - constant time

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

$O(n^3)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB
- costA - constant time
- costB - sum up the costs of subproblems (linear number)

Optimal strategy - baseline algorithm

Top-down algorithm with memoization.

Remember the path that minimizes the cost.

$O(n^2)$ space complexity

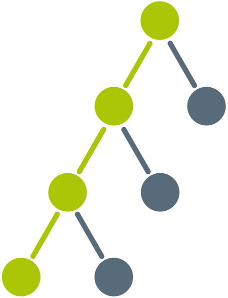
$O(n^3)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB
- costA - constant time
- costB - sum up the costs of subproblems (linear number)

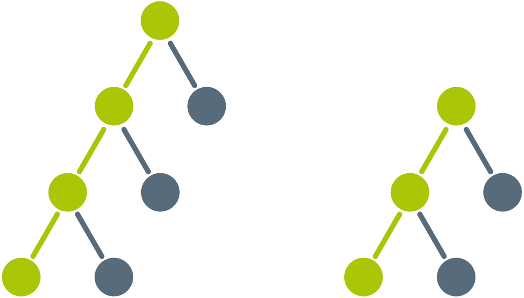
We need faster solution.

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Optimal strategy - baseline algorithm

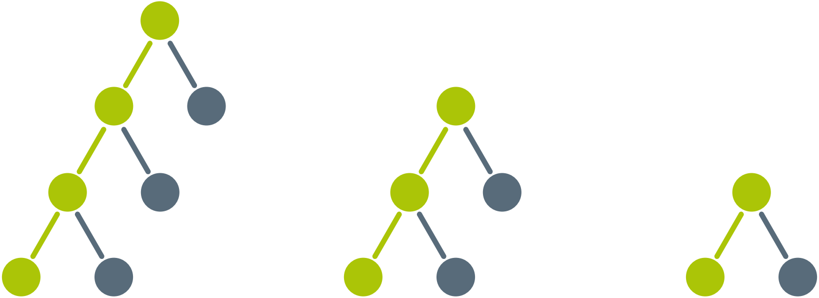


Optimal strategy - baseline algorithm

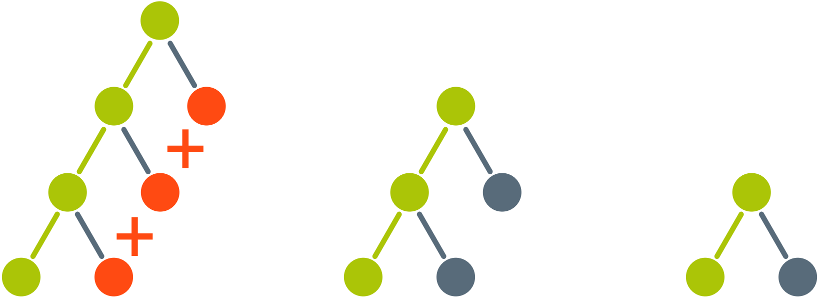


Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

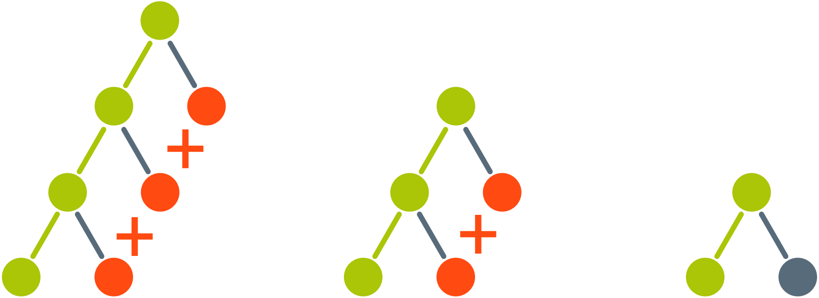
Optimal strategy - baseline algorithm



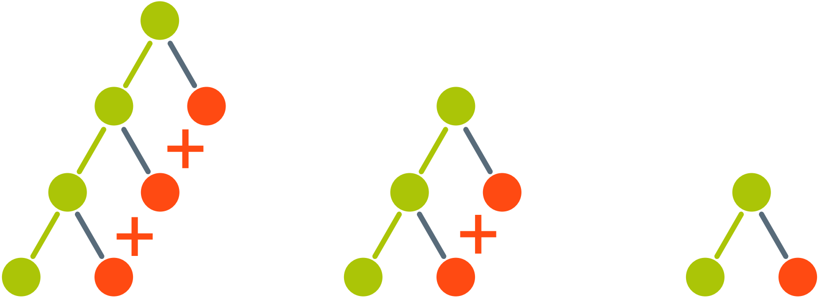
Optimal strategy - baseline algorithm



Optimal strategy - baseline algorithm



Optimal strategy - baseline algorithm



Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

$O(n^2)$ time complexity

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

$O(n^2)$ time complexity

- cost of each problem computed once (quadratic number)

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

$O(n^2)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

$O(n^2)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB
- costA - constant time

Optimal strategy - efficient algorithm

Bottom-up algorithm with dynamic programming.

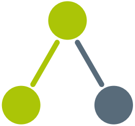
Order the problems to incrementally compute costB.

$O(n^2)$ space complexity

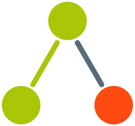
$O(n^2)$ time complexity

- cost of each problem computed once (quadratic number)
- for each problem evaluate costA + costB
- costA - constant time
- costB - incrementally computed in constant time - we add only one summand

Optimal strategy - efficient algorithm



Optimal strategy - efficient algorithm

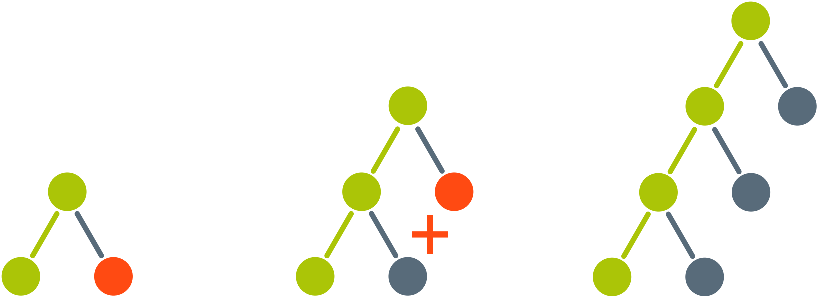


Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Optimal strategy - efficient algorithm

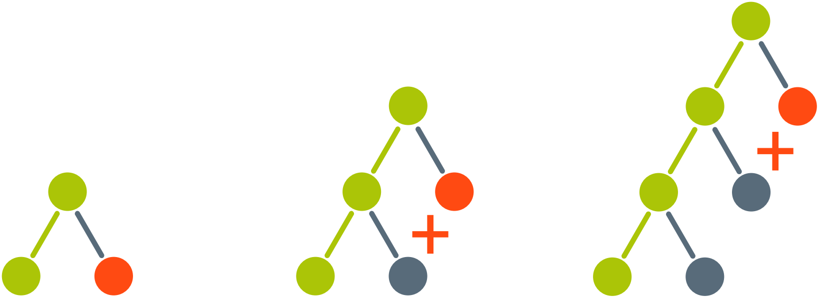
Optimal strategy - efficient algorithm

Optimal strategy - efficient algorithm



Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Optimal strategy - efficient algorithm



Experiments setup

Algorithms:

- Zhang-L only left paths in one tree
- Zhang-R only right paths in one tree
- Klein-H only heavy paths in one tree
- Demaine-H only heavy paths in bigger tree
- RTED optimal strategy

Experiments setup

Algorithms:

- Zhang-L only left paths in one tree
- Zhang-R only right paths in one tree
- Klein-H only heavy paths in one tree
- Demaine-H only heavy paths in bigger tree
- RTED optimal strategy

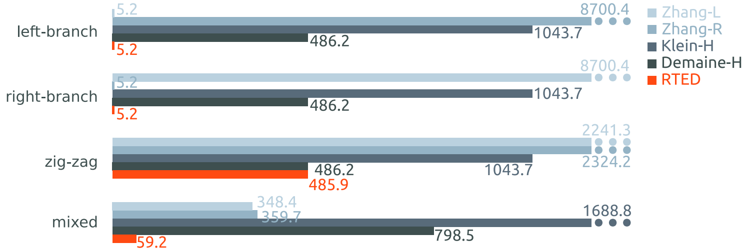
Tree shapes:

left-branch (Zhang-L) | right-branch (Zhang-R) | zig-zag (Demaine-H) | mixed

Scalability of distance computation

synthetic data:

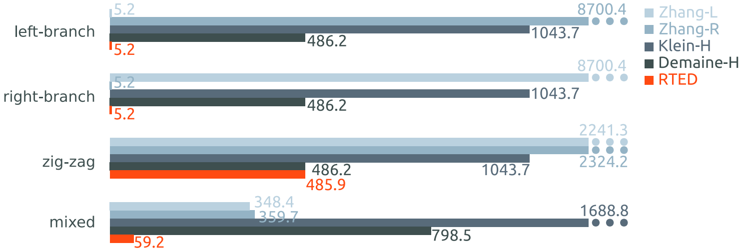
#subproblems$\times 10^6$ for trees of size $\sim$1000 nodes



Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Scalability of distance computation

synthetic data:
#subproblems$\times 10^6$ for trees of size $\sim$1000 nodes

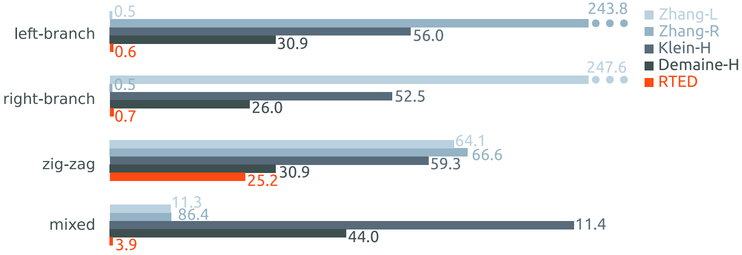


TreeFam (avg depth 14, max depth 158, avg size 95):
#subproblems to the best and worst competitor

- 86.9% of the best competitor
- 5.6% of the worst competitor

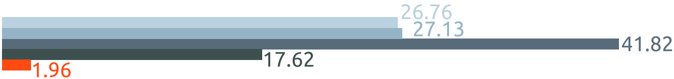
Scalability of distance computation

synthetic data:
runtime in seconds for trees of size ∼1000 nodes



Self similarity join

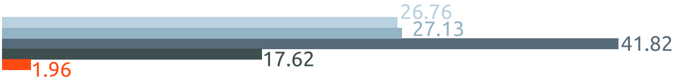
#subproblems$\times 10^9$ for trees of size $\sim$1000 nodes



RTED 9-21 times less subproblems

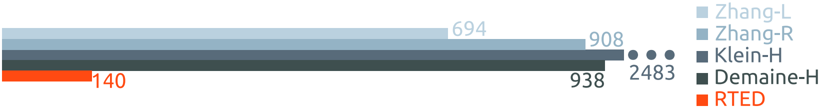
Self similarity join

#subproblems×10^9 for trees of size ∼1000 nodes



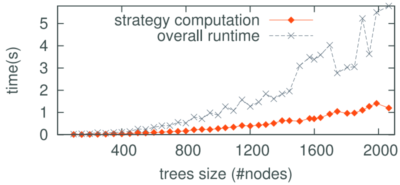
RTED 9-21 times less subproblems

runtime in seconds for trees of size ∼1000 nodes



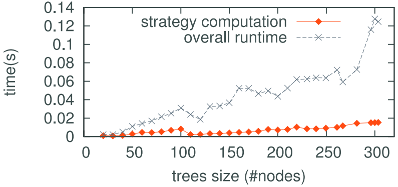
RTED 5-17 times faster

Overhead of strategy computation



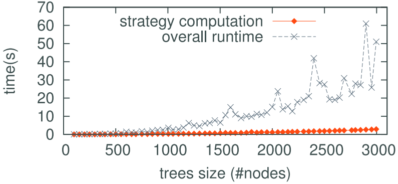
SwissProt

(max depth 4, max fanout 346, avg size 187)



TreeBank

(avg depth 10.4, max depth 35, avg size 68)



synthetic random trees

(varying size, fanout, depth)

Conclusion

First experimental evaluation of tree edit distance algorithms.

Conclusion

First experimental evaluation of tree edit distance algorithms.

State-of-the-art algorithms for tree edit distance

- tree shape matters
- choosing algorithms is hard
- wrong choice = large runtime difference

Conclusion

First experimental evaluation of tree edit distance algorithms.

State-of-the-art algorithms for tree edit distance

- tree shape matters
- choosing algorithms is hard
- wrong choice = large runtime difference

RTED

- uses best possible strategy
- insensitive to tree shape
- theoretical results and experiments prove its efficiency

Conclusion

First experimental evaluation of tree edit distance algorithms.

State-of-the-art algorithms for tree edit distance

- tree shape matters
- choosing algorithms is hard
- wrong choice = large runtime difference

RTED

- uses best possible strategy
- insensitive to tree shape
- theoretical results and experiments prove its efficiency

Open Source release

- http://www.inf.unibz.it/dis/projects/tree-distance-repository/

Future work

Find the mapping of the nodes.

- Tree edit distance is a number.
- In some scenarios the edit sequence is of interest.