Comparison and Construction of Phylogenetic Trees and Networks

Konstantinos Mampentzidis

PhD Defense

Aarhus University, Aarhus, Denmark

24 October 2019

AARHUS
UNIVERSITY

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Publications

- Gerth Stølting Brodal and Konstantinos Mampentzidis.
  Cache Oblivious Algorithms for Computing the Triplet Distance between Trees.
  In *ESA 2017*, Vienna, Austria.

- Jesper Jansson, Konstantinos Mampentzidis, Ramesh Rajaby, and Wing-Kin Sung.
  Computing the Rooted Triplet Distance Between Phylogenetic Networks.
  In *IWOCA 2019*, Pisa, Italy.

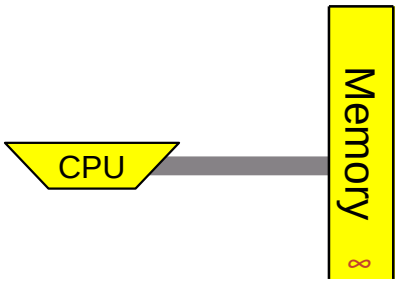
- Jesper Jansson, Konstantinos Mampentzidis, and Sandhya Thekkumpadan Puthiyaveedu.
  Building a Small and Informative Phylogenetic Supertree.
  In *WABI 2019*, Niagara Falls, USA.

Algorithmic Theory and Practice

- **Algorithm**: sequence of steps for solving a computational problem
- **Theory**: algorithms are first designed & analyzed in a **model of computation**
- **Practice**: then implemented in a programming language (C, C++, python, …)

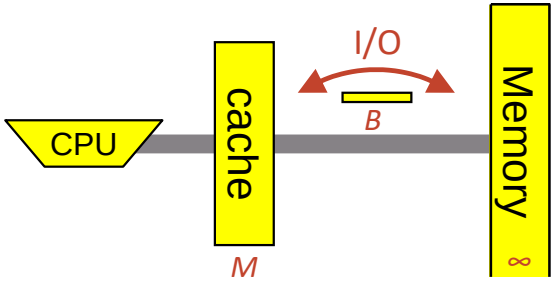
RAM model

John von Neumann 1945



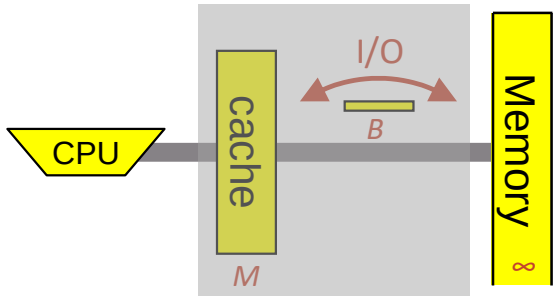
I/O model

Aggarwal and Vitter 1988



Cache Oblivious model

Frigo, Leiserson, Prokop, Ramachandran 1999

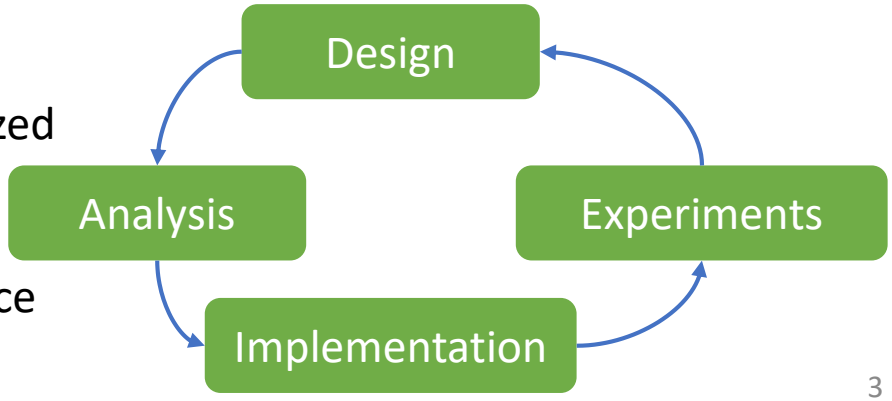


Computer architecture continues becoming more complicated

Gap between Theory and Practice

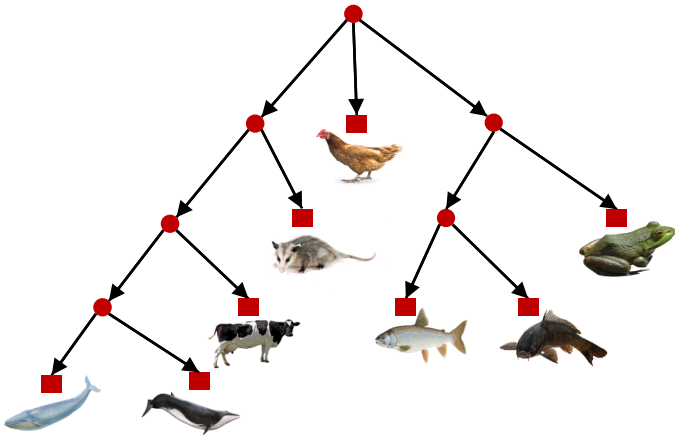
Algorithm Engineering

- Term first used by G. F. Italiano who organized the "Workshop on Algorithm Engineering" in Venice, Italy, 1997
- bridges the gap between theory and practice



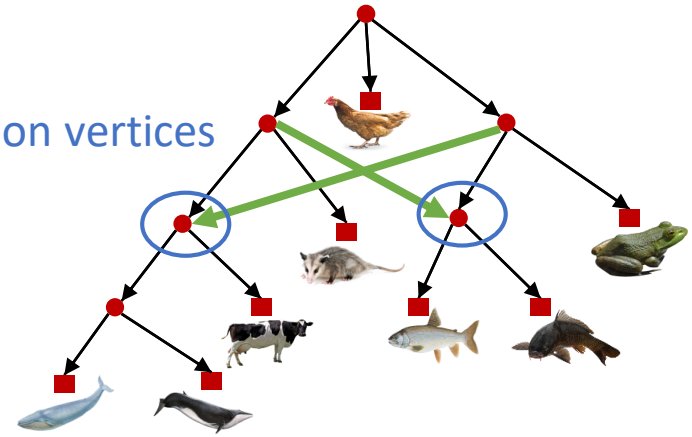
Problems in Phylogenetics

Rooted Phylogenetic Tree



Rooted Phylogenetic Network (DAG)

Reticulation vertices



- Different available data/construction algorithms can lead to trees/networks that look different
- Quantifying this difference can improve evolutionary inferences

ESA 2017

Given two rooted phylogenetic **trees** T_1 and T_2 over n species, how different are they?

IWOCA 2019

Given two rooted phylogenetic **networks** N_1 and N_2 over n species, how different are they?

- How are the trees and networks created to begin with?

WABI 2019

Given an input set of biological data, build a rooted phylogenetic **tree** that best represents it

Publications

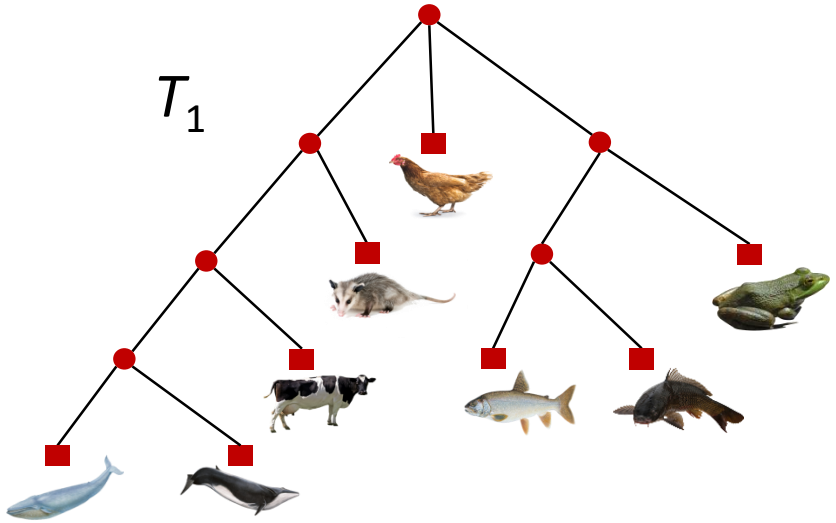
- **Gerth Stølting Brodal and Konstantinos Mampentzidis.**
  Cache Oblivious Algorithms for Computing the Triplet Distance between Trees.
  In *ESA 2017*, Vienna, Austria.

- Jesper Jansson, Konstantinos Mampentzidis, Ramesh Rajaby, and Wing-Kin Sung.
  Computing the Rooted Triplet Distance Between Phylogenetic Networks.
  In *IWOCA 2019*, Pisa, Italy.

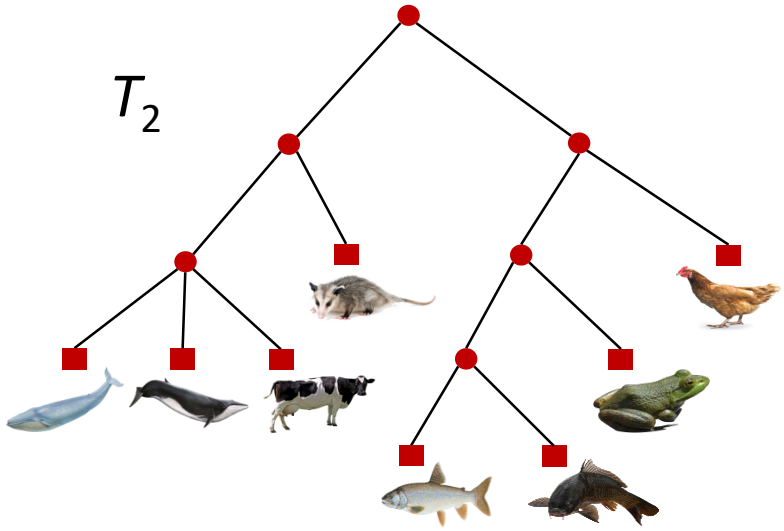
- Jesper Jansson, Konstantinos Mampentzidis, and Sandhya Thekkumpadan Puthiyaveedu.
  Building a Small and Informative Phylogenetic Supertree.
  In *WABI 2019*, Niagara Falls, USA.

Comparing Phylogenetic Trees

Rooted Phylogenetic Tree



Rooted Phylogenetic Tree



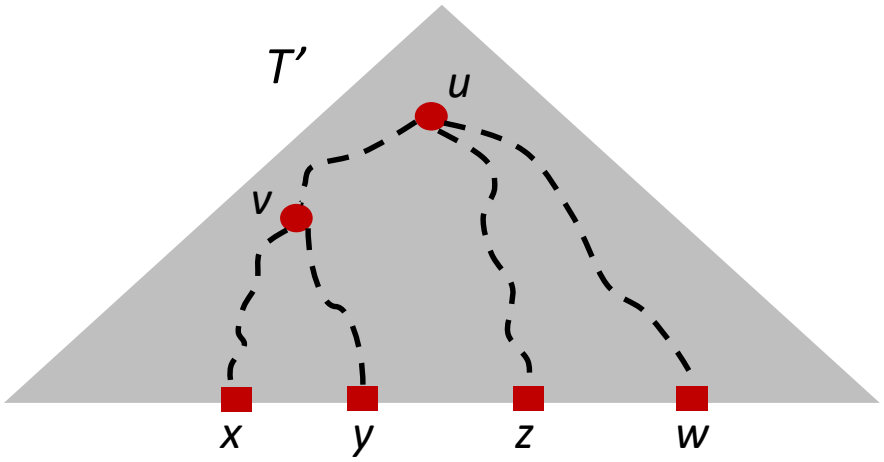
QUESTION

Given two rooted phylogenetic trees T_1 and T_2 over n species, how different are they?

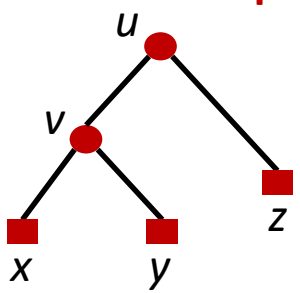
- Tree types: **rooted**/unrooted, **binary**/**arbitrary degree *d***
- Distance measures: **rooted triplet distance**, unrooted quartet distance, Robinson-Foulds, …

Rooted Triplet Distance (Trees)

- A **rooted triplet** is defined by 3 leaf labels and their induced tree topology
- A triplet is **induced by a tree** T' if it appears as an embedded subtree in T'

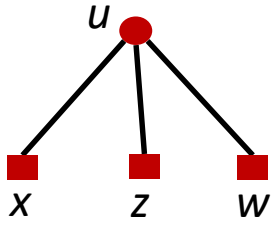


Resolved triplet



$xy|z$

Fan triplet



$x|z|w$

Rooted Triplet Distance (Trees), Dobson [Combinatorial Mathematics III 1975]

Let T_1 and T_2 be two rooted trees built on the same leaf label set Λ of size n

Shared triplets = triplets that are induced by both T_1 and T_2

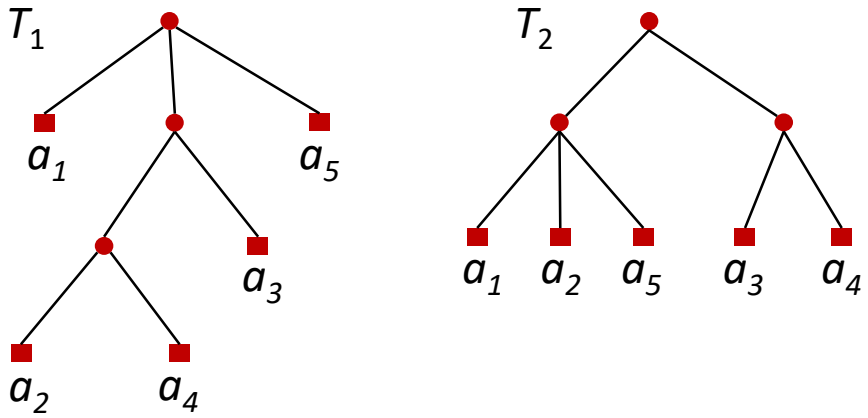
$S(T_1, T_2)$ = # shared triplets $\leq \binom{n}{3}$

Rooted triplet distance $D(T_1, T_2) = \binom{n}{3} - S(T_1, T_2)$ = # non-shared triplets

Rooted Triplet Distance (Trees)

Rooted Triplet Distance (Trees), Dobson [Combinatorial Mathematics III 1975]
Let T_1 and T_2 be two rooted trees built on the same leaf label set Λ of size n
Shared triplets = triplets that are induced by both T_1 and T_2
$S(T_1, T_2)$ = # shared triplets $\leq \binom{n}{3}$
Rooted triplet distance $D(T_1, T_2) = \binom{n}{3} - S(T_1, T_2)$ = # non-shared triplets

Example



shared triplets

$a_3a_4|a_5$

$a_3a_4|a_1$

$a_1|a_2|a_5$

non-shared triplets

a_1, a_2, a_3 a_2, a_3, a_5

a_1, a_3, a_5 a_2, a_4, a_3

a_1, a_2, a_4 a_2, a_4, a_5

a_1, a_4, a_5

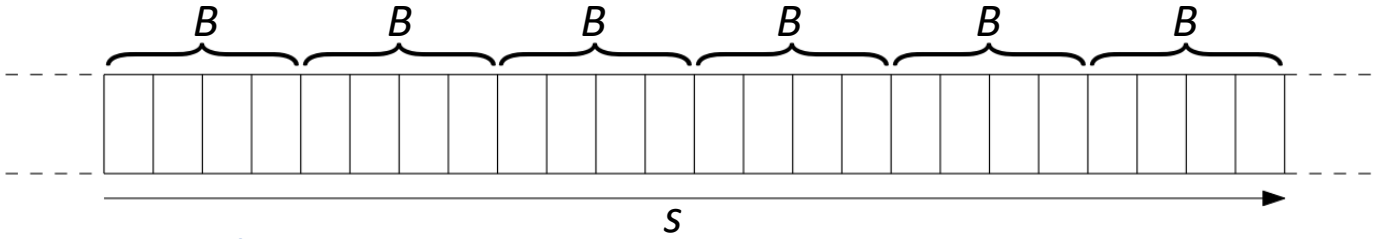
$D(T_1, T_2) = 7$

Previous and New Results

Reference	Time	I/Os	Space	Non-Binary Trees
Critchlow *et al.* [Sys. Biology 1996]	$O(n^2)$	$O(n^2)$	$O(n^2)$	no
Bansal *et al.* [TCS 2011]	$O(n^2)$	$O(n^2)$	$O(n^2)$	yes
Sand *et al.* [BMC Bioinform. 2013] ★	$O(n \cdot \log^2 n)$	$O(n \cdot \log^2 n)$	$O(n)$	no
Brodal *et al.* [SODA 2013] ★	$O(n \cdot \log n)$	$O(n \cdot \log n)$	$O(n \cdot \log n)$	yes
Jansson & Rajaby [JCB 2017] ★	$O(n \cdot \log^3 n)$	$O(n \cdot \log^3 n)$	$O(n \cdot \log n)$	yes
new [ESA 2017] ★	$O(n \cdot \log n)$	$O(n/B \cdot \log_2(n/M))$	$O(n)$	yes

★ **Implementation available**

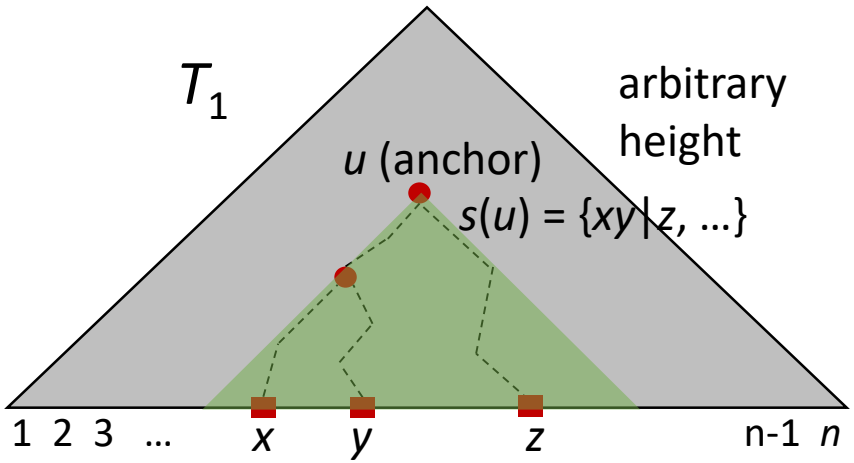
- All previous solutions rely heavily on random memory access
  - Penalized by cache performance
  - Do not scale to external memory
- The new algorithms rely on *scanning* continuous chunks of memory
  - Scanning s elements requires $O(s/B)$ I/Os in the cache oblivious model

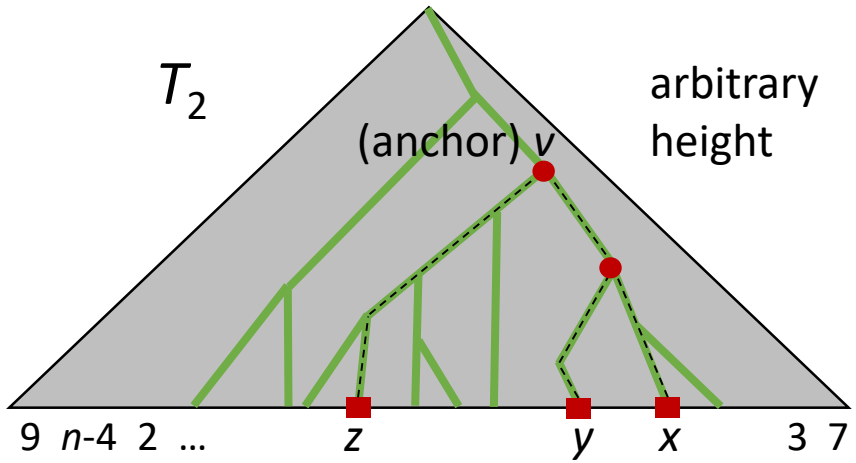


  - Scale to external memory

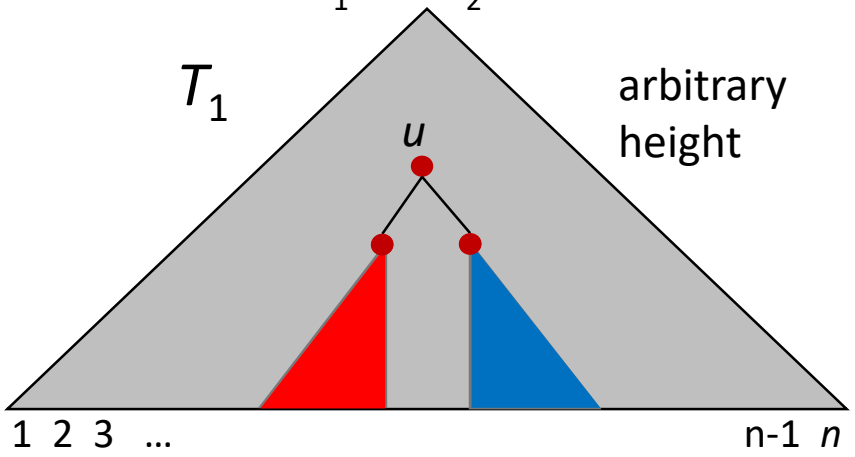
Previous Approaches – Quadratic Algorithm

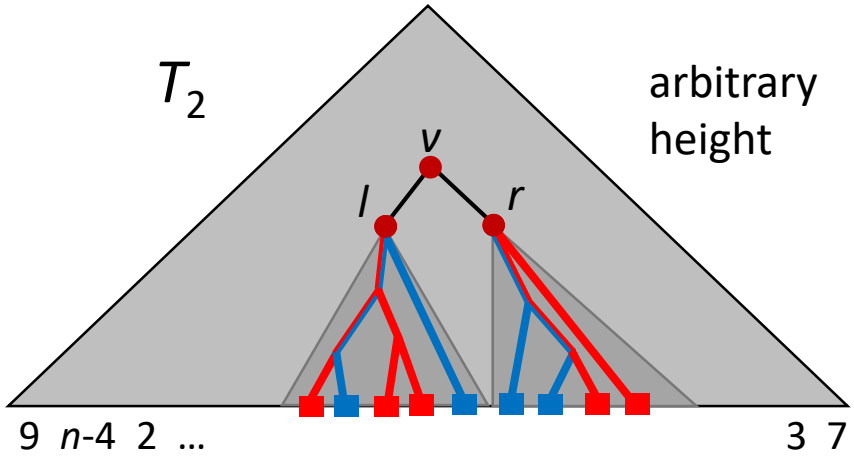
- Basis for all O(n·polylog n) results: O(n^2) algorithm for binary trees in [BMC Bioinform. 2013]





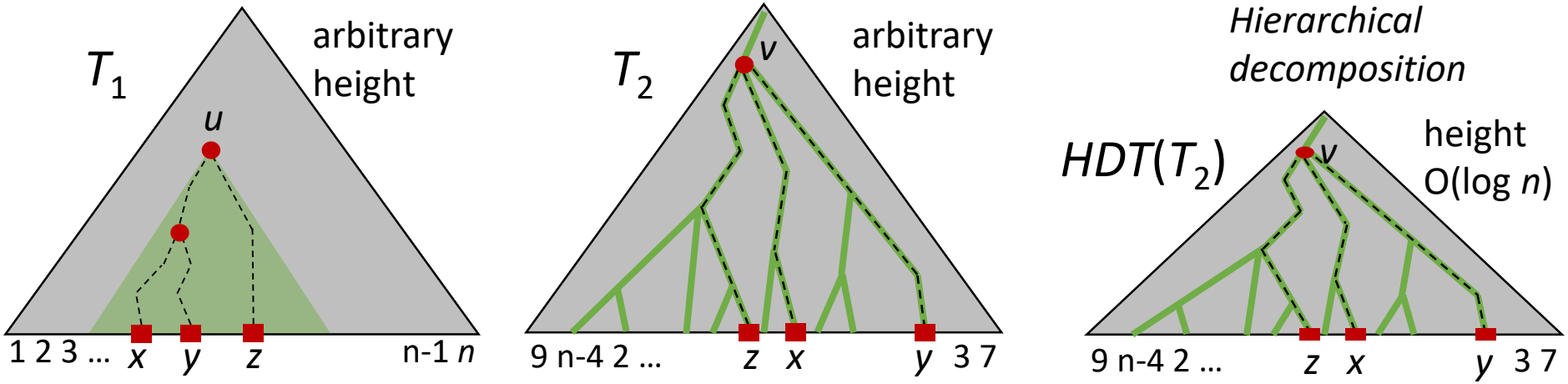
- Every triplet with leaves x, y, and z is anchored in $LCA(x, y, z)$ (anchor node)
- $s(u)$: set containing all triplets anchored in u
- $S(T_1, T_2) = \sum_{u \in T_1} \sum_{v \in T_2} |s(u) \cap s(v)|$





$$|s(u) \cap s(v)| = \binom{l_{\text{red}}}{2} r_{\text{blue}} + \binom{l_{\text{blue}}}{2} r_{\text{red}} + \binom{r_{\text{red}}}{2} l_{\text{blue}} + \binom{r_{\text{blue}}}{2} l_{\text{red}}$$

Previous Approaches – Subquadratic Algorithms



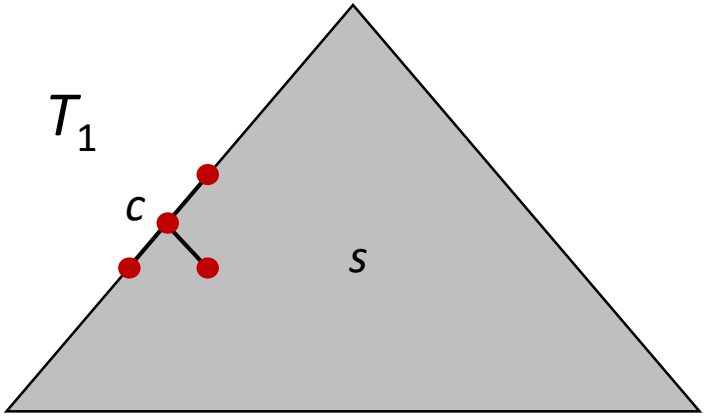
- For $u \in T_1$ the $HDT(T_2)$ maintains $\sum_{v \in T_2} |s(u) \cap s(v)|$
- Each leaf color change in T_1 yields an update to $HDT(T_2)$

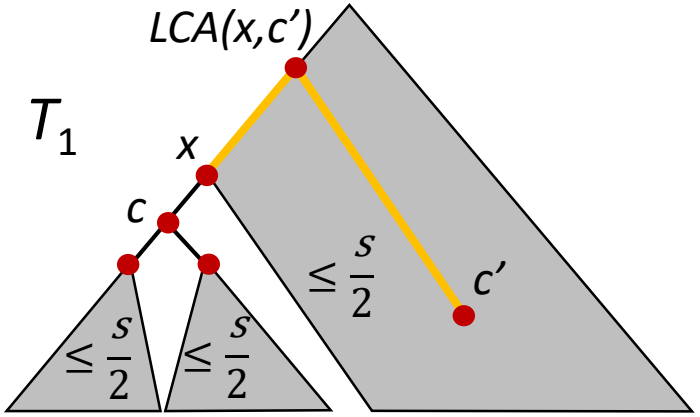
$\Theta(n \log n)$ updates, with each update corresponding to a leaf to root path traversal of $HDT(T_2)$ **Bad I/O performance**

Reference	Time	$HDT(T_2)$
Sand *et al.* [BMC Bioinform. 2013]	$O(n \cdot \log^2 n)$	Static
Brodal *et al.* [SODA 2013]	$O(n \cdot \log n)$	Dynamic/Contraction
Jansson & Rajaby [JCB 2017]	$O(n \cdot \log^3 n)$	Static (heavy-light decomposition)

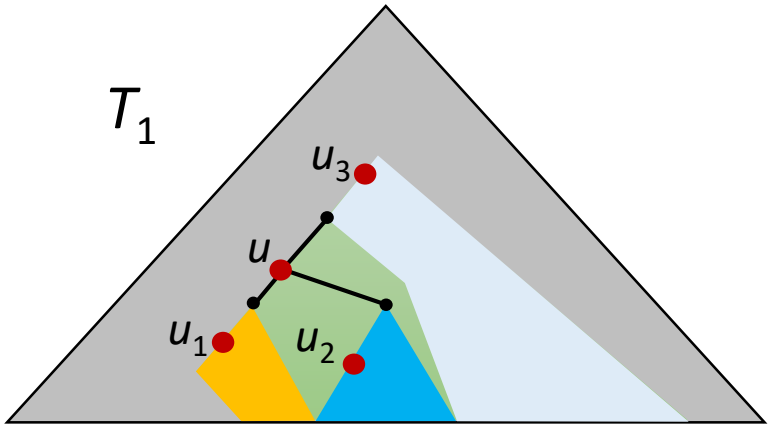
The New Algorithm for Binary Trees (ESA 2017)

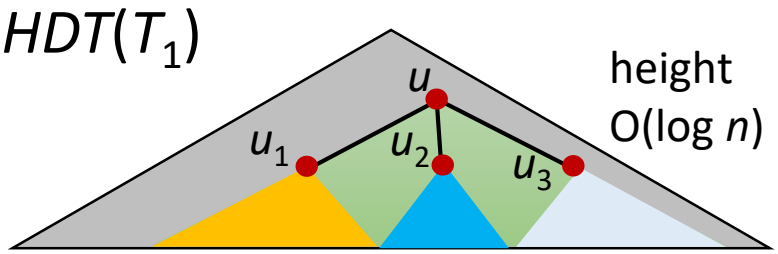
- New order of visiting nodes of T_1 based on DFS traversal of an $HDT(T_1)$
- $HDT(T_1)$ = ***modified centroid decomposition***





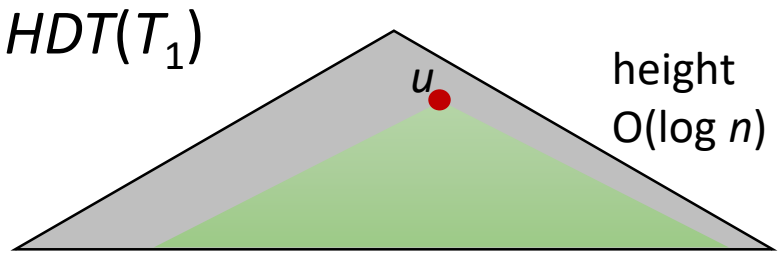
- **Lemma 2** height($HDT(T_1)$) ≤ 2 + 2·log s = O(log n)

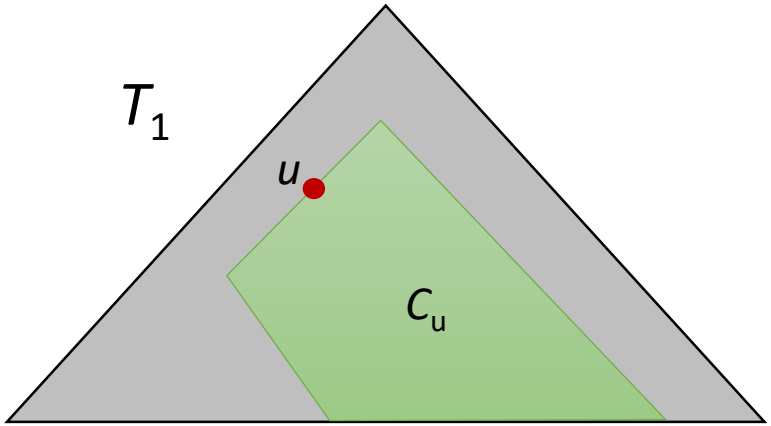


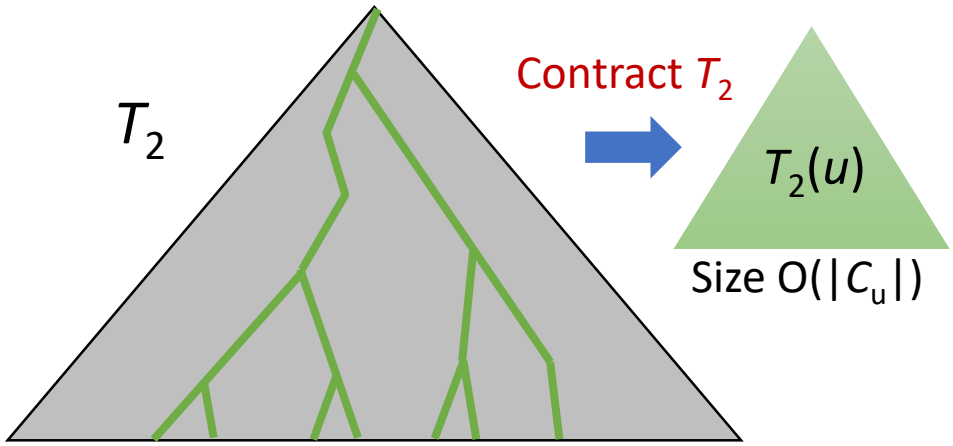


- **Order to visit the nodes in T_1:** DFS traversal of $HDT(T_1)$, where the children of a node u are visited from left to right

The New Algorithm for Binary Trees (ESA 2017)





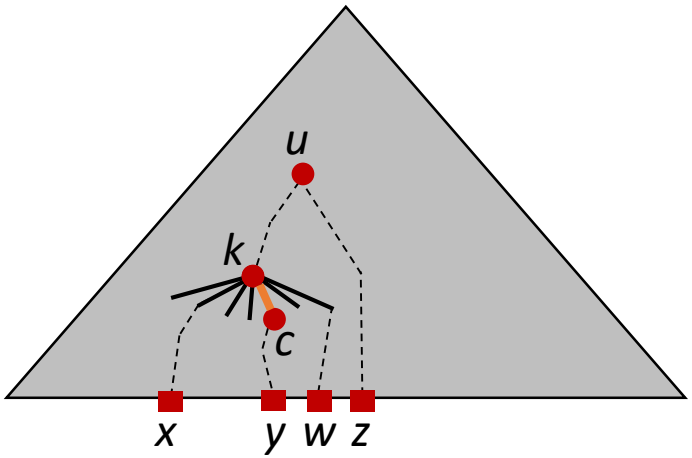


For every node u in $HDT(T_1)$ we scan $T_2(u)$ to count $\sum_{v \in T_2} |s(u) \cap s(v)|$

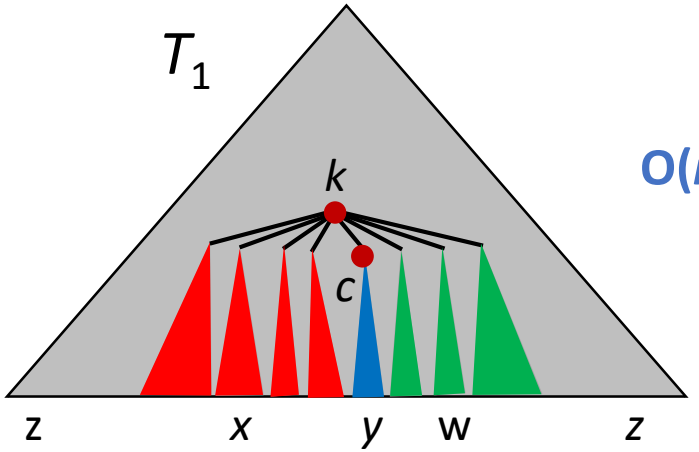
- **RAM model:** O(n) time per level of $HDT(T_1)$ → O($n \cdot \log n$)
- **To scale to external memory:** store every component/contracted tree in memory following a proper layout such that scanning a component/contracted tree of size s takes O(s/B) I/Os

The New Algorithm for General Trees (ESA 2017)

1. Anchor triplets in edges instead of nodes

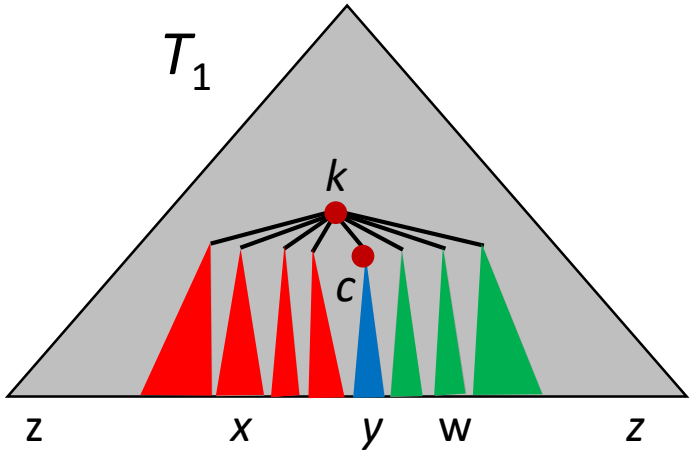


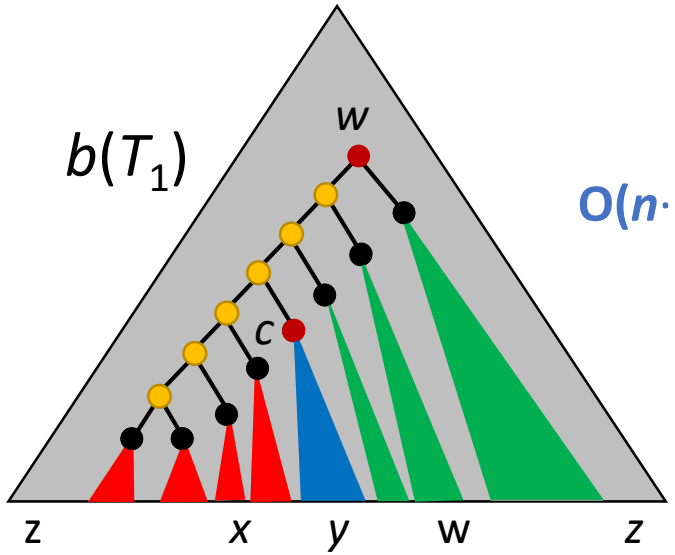
2. Capture triplets with 4 colors



O(n^2)

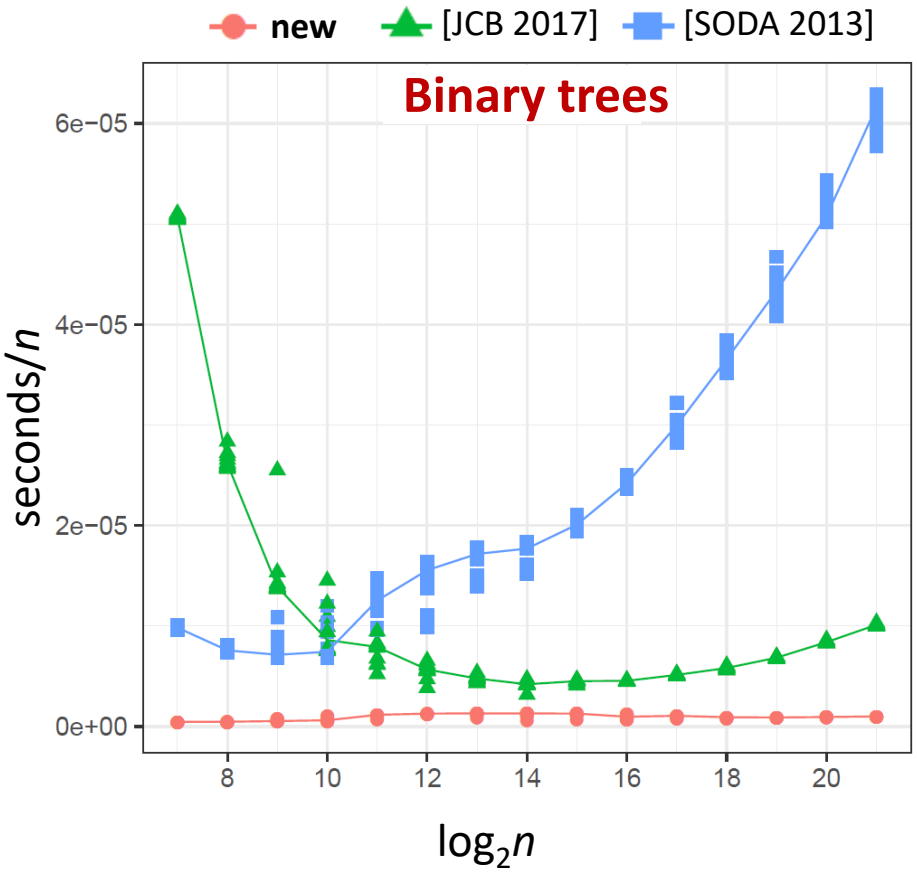
3. Transform T_1 into a binary tree $b(T_1)$

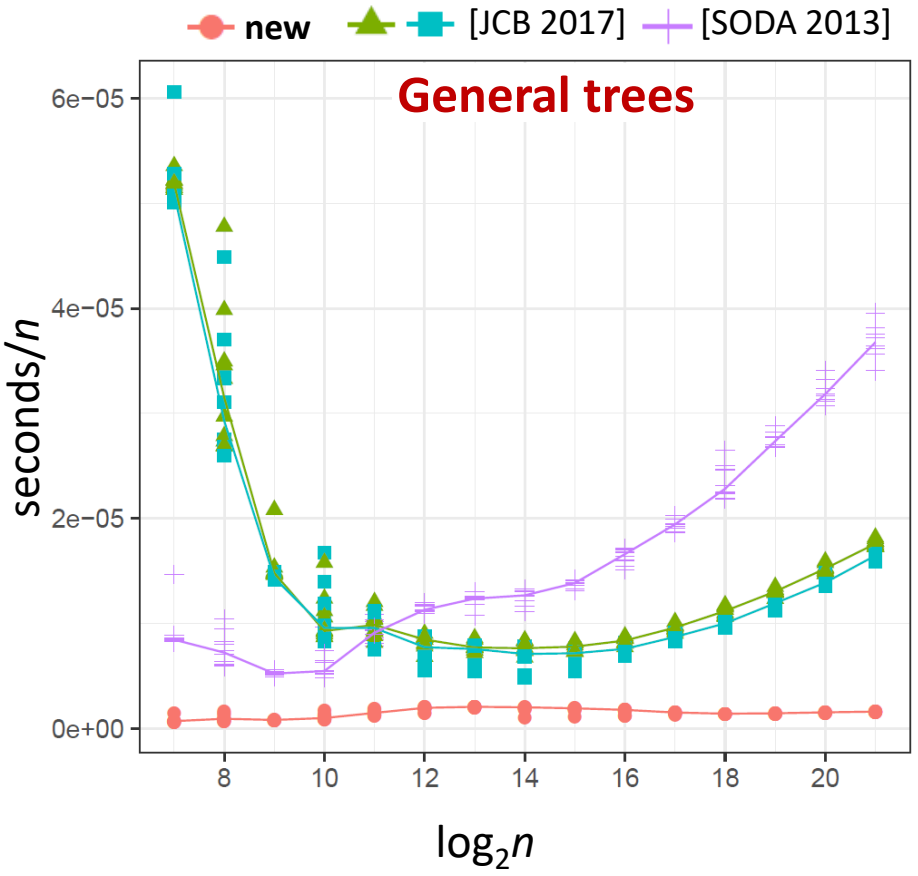




O(n·log n)

RAM Experiments – Time Performance





Source code: https://github.com/kmampent/CacheTD

I/O Experiments – Time Performance

Binary Trees

n	[JCB 2017] Previous best	[SODA 2013]	**New**
2^{15}	1s	1s	1s
2^{16}	1s	2s	1s
2^{17}	1s	4s	1s
2^{18}	2s	1m:03s	1s
2^{19}	4s	1h:21m	1s
2^{20}	9s	≥ 10h	1s
2^{21}	13m:12s	✗	3s
2^{22}	≥ 10h	✗	9s
2^{23}	✗	✗	3m:37s
2^{24}	✗	✗	10m:35s

General Trees

n	[JCB 2017] Previous best	[SODA 2013]	**New**
2^{15}	1s	1s	1s
2^{16}	1s	1s	1s
2^{17}	1s	3s	1s
2^{18}	3s	7s	1s
2^{19}	7s	5m:20s	1s
2^{20}	3m:43s	≥ 10h	2s
2^{21}	≥ 10h	✗	20s
2^{22}	✗	✗	2m:02s
2^{23}	✗	✗	10m:42s
2^{24}	✗	✗	42m:06s

Source code: https://github.com/kmampent/CacheTD

Publications

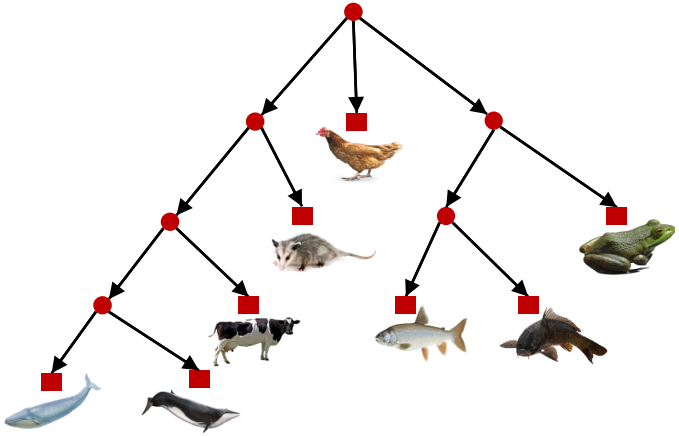
- Gerth Stølting Brodal and Konstantinos Mampentzidis.
  Cache Oblivious Algorithms for Computing the Triplet Distance between Trees.
  In *ESA 2017*, Vienna, Austria.

- **Jesper Jansson, Konstantinos Mampentzidis, Ramesh Rajaby, and Wing-Kin Sung.**
  Computing the Rooted Triplet Distance Between Phylogenetic Networks.
  In *IWOCA 2019*, Pisa, Italy.

- Jesper Jansson, Konstantinos Mampentzidis, and Sandhya Thekkumpadan Puthiyaveedu.
  Building a Small and Informative Phylogenetic Supertree.
  In *WABI 2019*, Niagara Falls, USA.

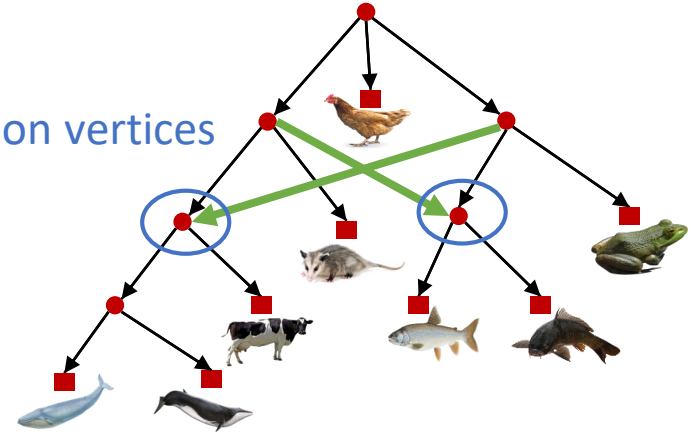
Rooted Phylogenetic Networks

Rooted Phylogenetic Tree



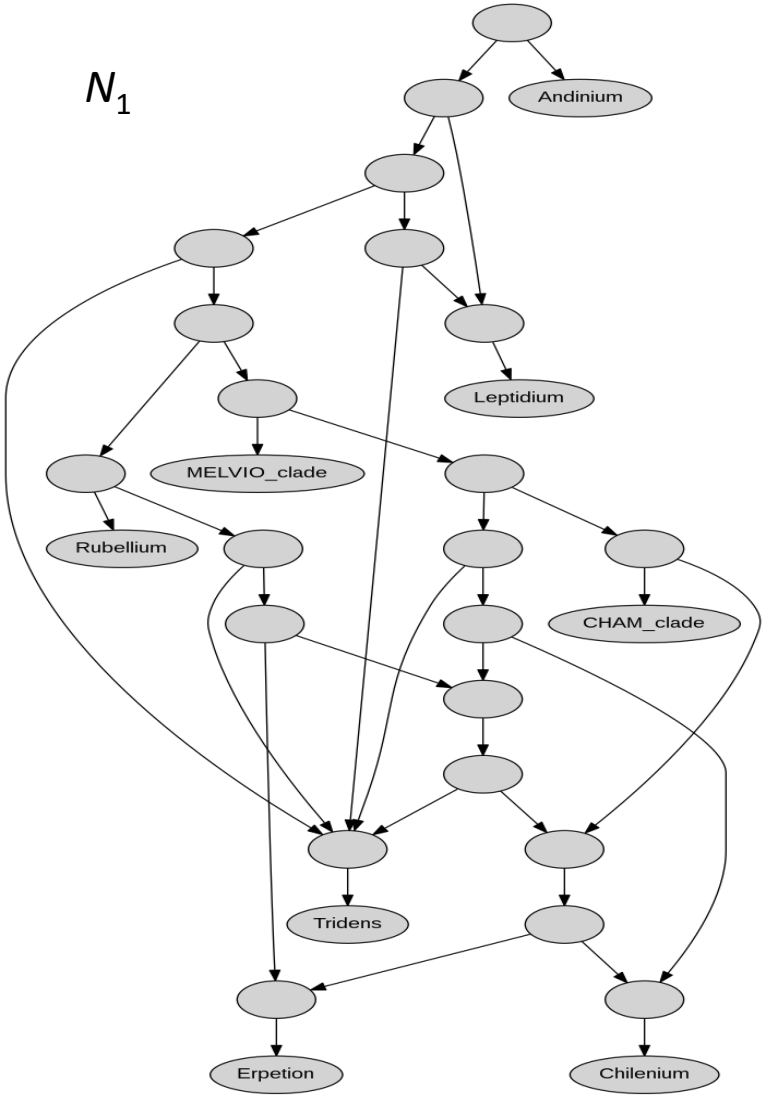
Rooted Phylogenetic Network (DAG)

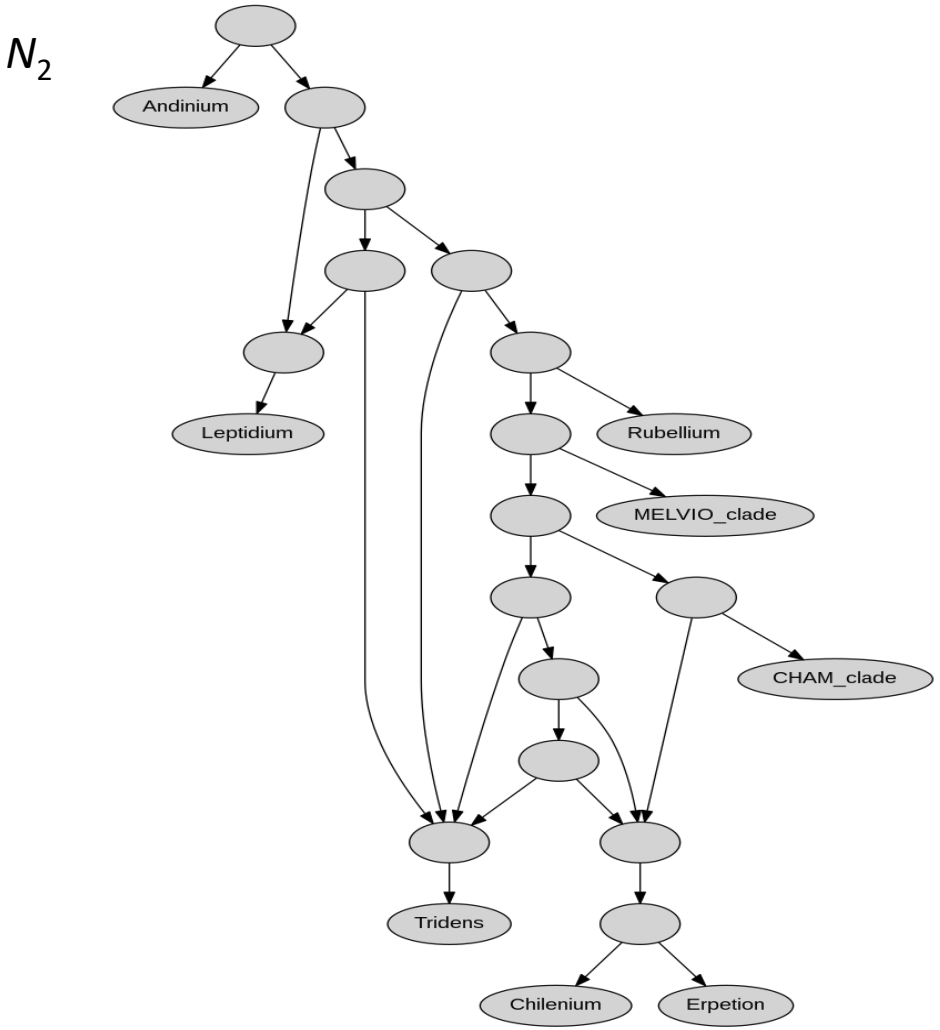
Reticulation vertices



An “example” of a hybrid animal

Rooted Phylogenetic Networks - Example

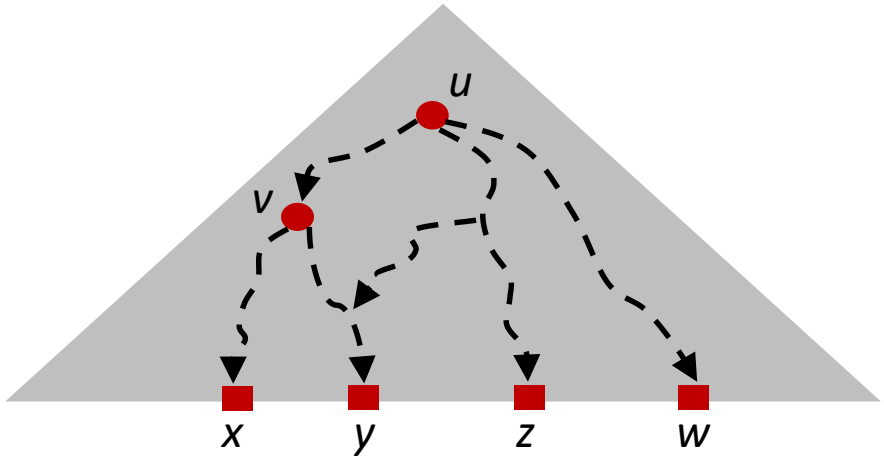




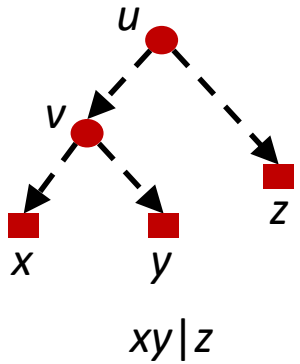
Marcussen *et al*. From gene trees to a dated allopolyploid network: insights from the angiosperm genus Viola (Violaceae). Systematic Biology 64 (1) (2015) 84–101

Rooted Triplet Distance - Networks

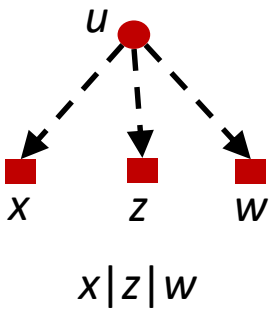
- Invented by Dobson for trees [Combinatorial Mathematics III 1975]
  3 leaves → unique tree topology
- Gambette and Huber extended it to networks [JMB 2012]
  3 leaves → one or more tree topologies
- A **rooted triplet** is defined by 3 leaf labels and their induced tree topology in the network



Resolved triplet



Fan triplet

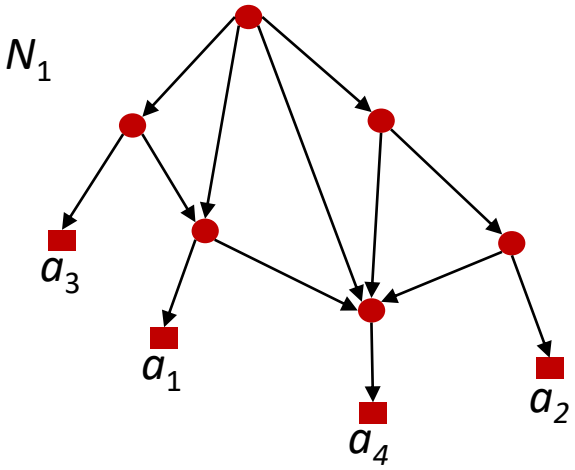


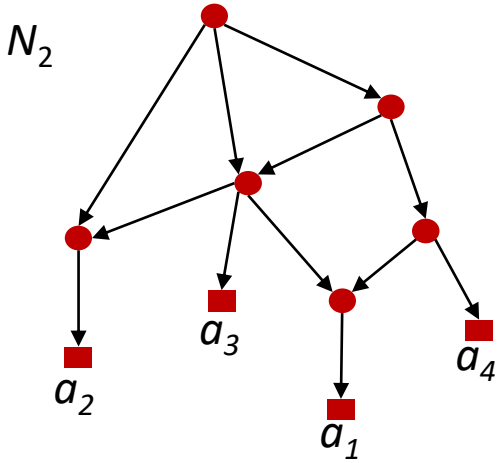
- *Shared triplets* = triplets that appear in both N_1 and N_2
- *Different triplets* = triplets that appear only in N_1 or only in N_2
- $S(N_1, N_2)$ = # shared triplets $\leq 4 \cdot \binom{n}{3}$
- *Rooted triplet distance* $D(N_1, N_2)$ = # different triplets = $S(N_1, N_1) + S(N_2, N_2) - 2 \cdot S(N_1, N_2)$

Rooted Triplet Distance - Networks

- *Shared triplets* = triplets that appear in both N_1 and N_2
- *Different triplets* = triplets that appear only in N_1 or only in N_2
- $S(N_1, N_2)$ = # shared triplets $\leq 4 \cdot \binom{n}{3}$
- *Rooted triplet distance* $D(N_1, N_2)$ = # different triplets = $S(N_1, N_1) + S(N_2, N_2) - 2 \cdot S(N_1, N_2)$

Example





shared triplets

$a_1|a_2|a_3$ $a_1a_3|a_4$

$a_1a_3|a_2$ $a_1a_4|a_3$

$a_1|a_2|a_4$ $a_2|a_3|a_4$

$a_1a_4|a_2$ $a_3a_4|a_2$

different triplets

$a_2a_3|a_1$ $a_2a_3|a_4$

$a_1|a_3|a_4$ $a_2a_4|a_3$

$a_2a_4|a_1$

$a_1a_2|a_4$

$D(N_1, N_2) = 6$

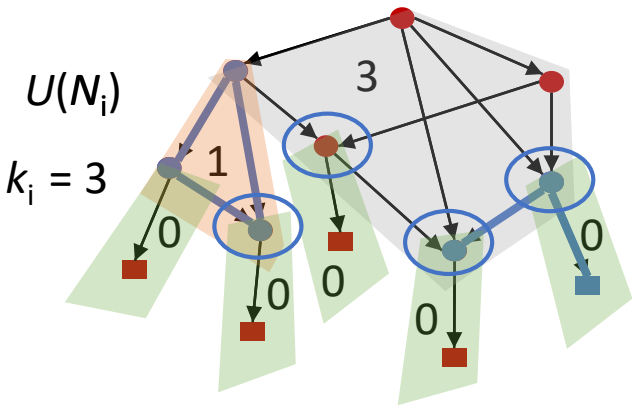
Previous and New Results

Reference	*k* (level)	Degrees	Time Complexity
Fortune *et al.* [TCS 1980]	arbitrary	arbitrary	$\Omega(N^7 n^3)$
Byrka *et al.* [JDA 2010]	arbitrary	binary	$O(N^3 + n^3)$
Byrka *et al.* [JDA 2010] ★	arbitrary	binary	$O(N + k^2 N + n^3)$
Brodal *et al.* [SODA 2013, ESA 2017] ★	0 (trees)	arbitrary	$O(n \cdot \log n)$
Jansson *et al.* [JCB 2019] ★	1 (galled trees)	arbitrary	$O(n \cdot \log n)$
new [IWOCA 2019] Algorithm I ★	arbitrary	arbitrary	$O(N^2 M + n^3)$
new [IWOCA 2019] Algorithm II ★	arbitrary	arbitrary	$O(M + k^3 d^3 n + n^3)$

★ **Implementation available**

- $N_1 = (V_1, E_1)$, $N_2 = (V_2, E_2)$, and n is the size of the common leaf label set
- d_1 = maximum in-degree of a vertex in N_1. Similarly, we have d_2 for N_2
- $N = \max(|V_1|, |V_2|)$, $M = \max(|E_1|, |E_2|)$, and $d = \max(d_1, d_2)$

k? Measures treelikeness



- A subgraph H of $U(N_i)$ is *biconnected* if it is not possible to remove exactly one vertex from H to make it disconnected
- A subgraph H' is a *biconnected component* of $U(N_i)$ if it is a maximal biconnected subgraph of $U(N_i)$
- N_i has level k_i if there are $\leq k_i$ reticulation vertices in any biconnected component of $U(N_i)$
- $k = \max(k_1, k_2)$

Previous and New Results

Reference	*k* (level)	Degrees	Time Complexity
Fortune *et al.* [TCS 1980]	arbitrary	arbitrary	$\Omega(N^7n^3)$
Byrka *et al.* [JDA 2010]	arbitrary	binary	$O(N^3 + n^3)$
Byrka *et al.* [JDA 2010]	arbitrary	binary	$O(N + k^2N + n^3)$
Brodal *et al.* [SODA 2013, ESA 2017] ★ ★	0 (trees)	arbitrary	$O(n \log n)$
Jansson *et al.* [JCB 2019] ★	1 (galled trees)	arbitrary	$O(n \log n)$
new [IWOCA 2019] Algorithm I ★	arbitrary	arbitrary	$O(N^2M + n^3)$
new [IWOCA 2019] Algorithm II ★	arbitrary	arbitrary	$O(M + k^3d^3n + n^3)$

★ **Implementation available**

- ***k* = 0 (trees), arbitrary degrees**

$O(n^2)$ [TCS 2011] $O(n \cdot \log n)$ [SODA 2013] $O(n \cdot \log^3 n)$ [JCB 2017] $O(n \cdot \log n)$ [ESA 2017]

fast in practice

scales to external memory

fastest in practice

- ***k* = 1 (galled trees), arbitrary degrees**

$O(n^{2.687})$ [JDA 2014] $O(n \cdot \log n)$ [JCB 2019]

count triangles in a graph

combine the outputs of an algorithm on O(1) instances when $k = 0$

- **arbitrary *k*, binary degrees**

$O(N^3 + n^3)$ and $O(N + k^2N + n^3)$ [JDA 2010]

Construct a data structure in $O(N^3)$ or $O(N + k^2N)$ time

Use it to test the consistency of any triplet in O(1) time

- **arbitrary *k*, arbitrary degrees**

$\Omega(N^7n^3)$ [TCS 1980]

Use pattern matching algorithm to test the consistency of a triplet in $\Omega(N^7)$ time

$O(N^2M + n^3)$ and $O(M + k^3d^3n + n^3)$ **[IWOCA 2019]**

Construct a data structure in $O(N^2M)$ or $O(M + k^3d^3n)$ time

Use it to test the consistency of any triplet in O(1) time

Algorithm I (IWOCA 2019)

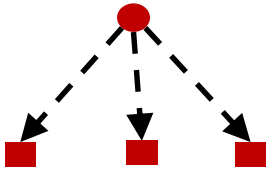
- We extend a technique by Shiloach and Perl [J. ACM 1973]

Problem
Input DAG $G = (V, E)$ and 4 vertices s_1, t_1, s_2, t_2
Output Are there two disjoint paths in G, one from s_1 to t_1 and one from s_2 to t_2?

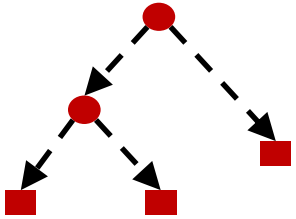
Solution
1. Build a DAG G' in $O(|V| \cdot |E|)$ time
2. Return ***TRUE*** if there exists a path from (s_1, s_2) to (t_1, t_2) in G', ***FALSE*** o/w

Our approach

Fan triplets



Resolved triplets

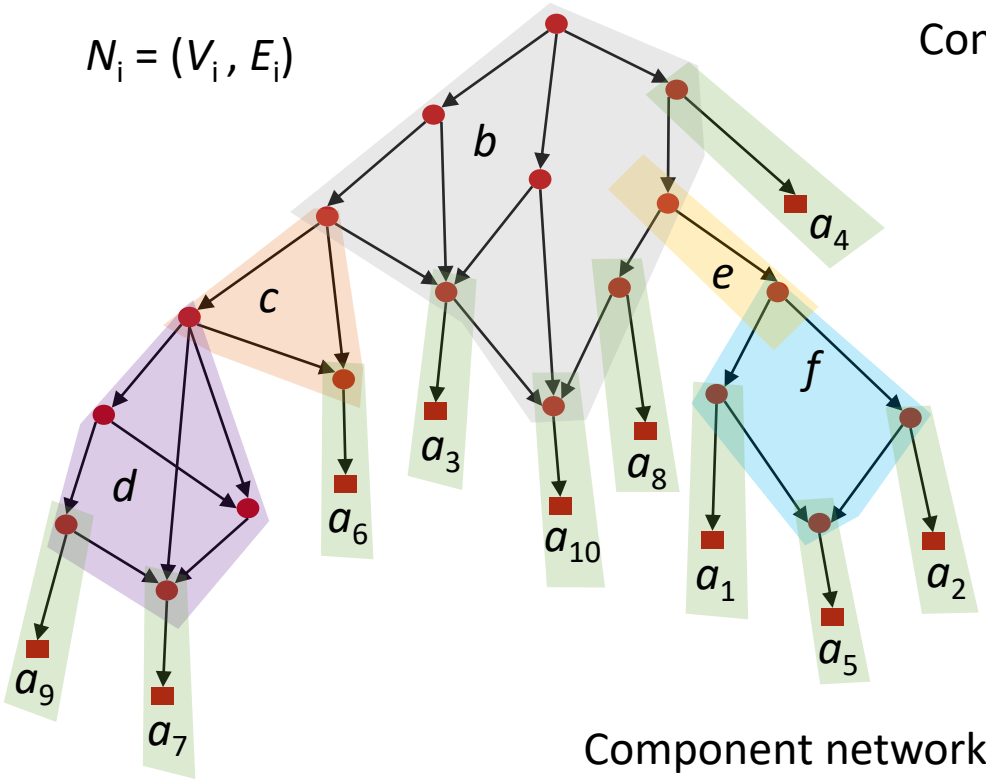


- For a network N_i we define a *fan graph* N_i^f and a *fan table* A_i^f $O(|V_i|^2 \cdot |E_i|)$
- We then use A_i^f to determine the consistency of any fan triplet with N_i in O(1) time

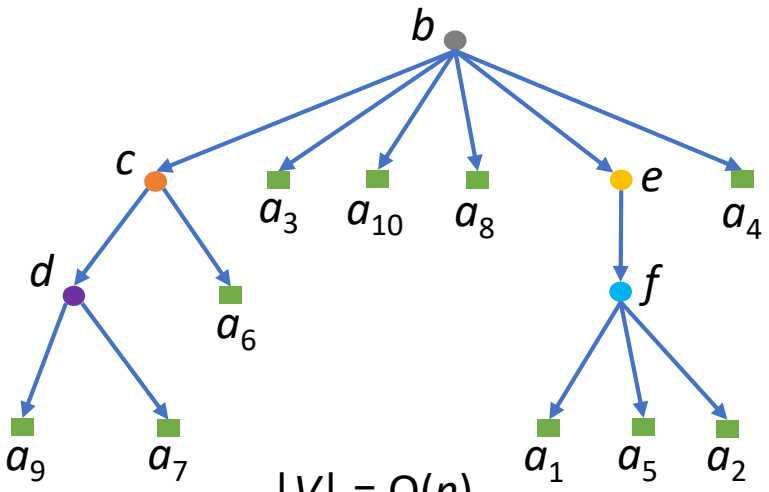
- For a network N_i we define a *resolved graph* N_i^r and a *resolved table* A_i^r $O(|V_i|^2 \cdot |E_i|)$
- We then use A_i^r to determine the consistency of any resolved triplet with N_i in O(1) time

Algorithm II (IWOCA 2019)

$N_i = (V_i, E_i)$

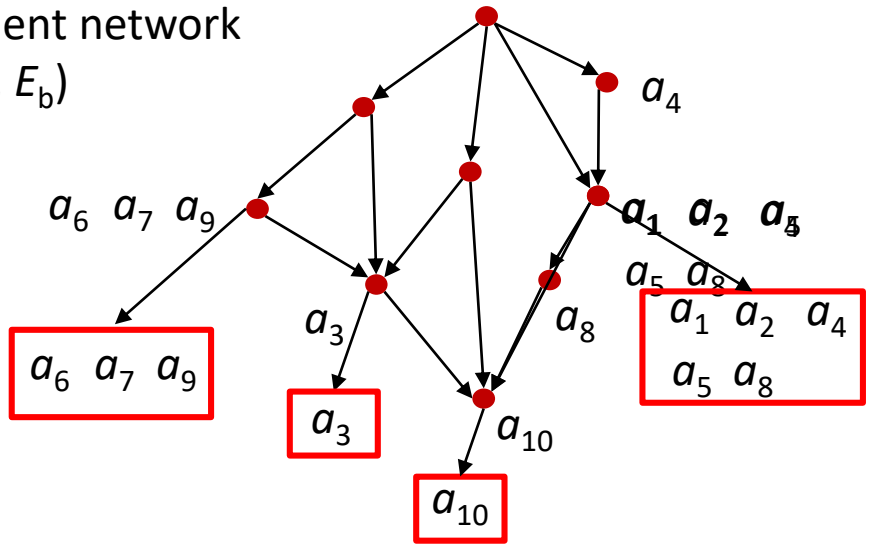


Component tree $T = (V, E)$



$|V| = O(n)$
$|E| = O(n)$

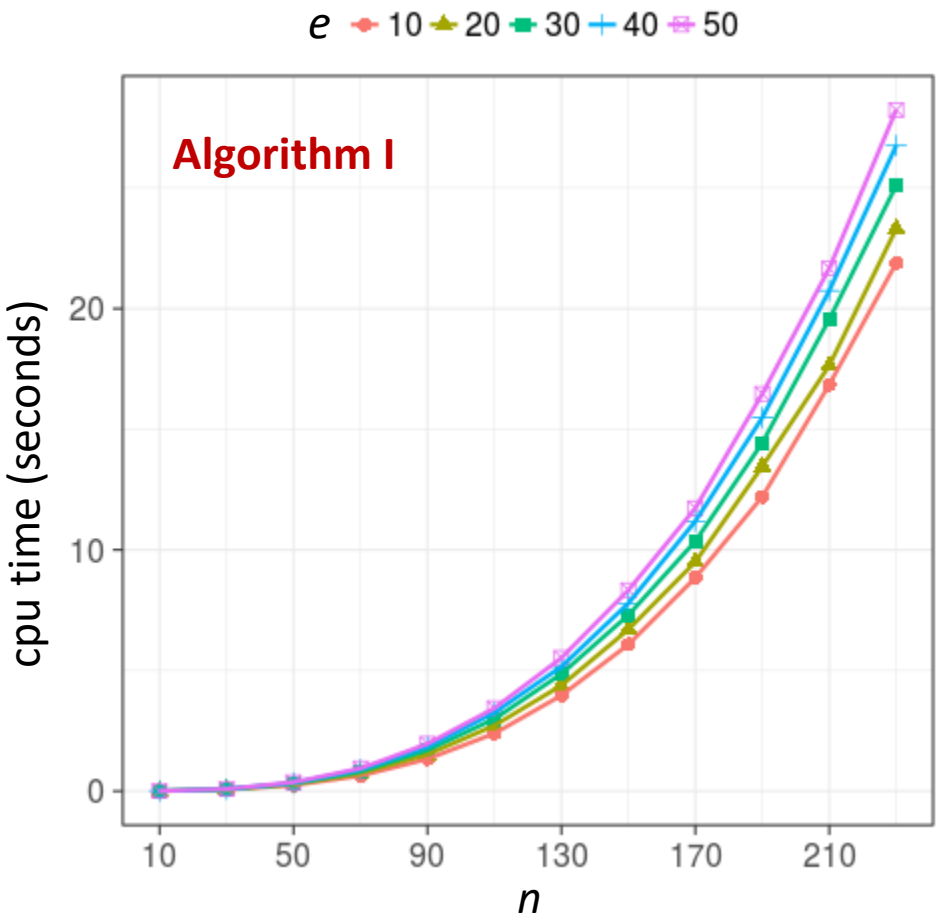
Component network
$C_b = (V_b, E_b)$

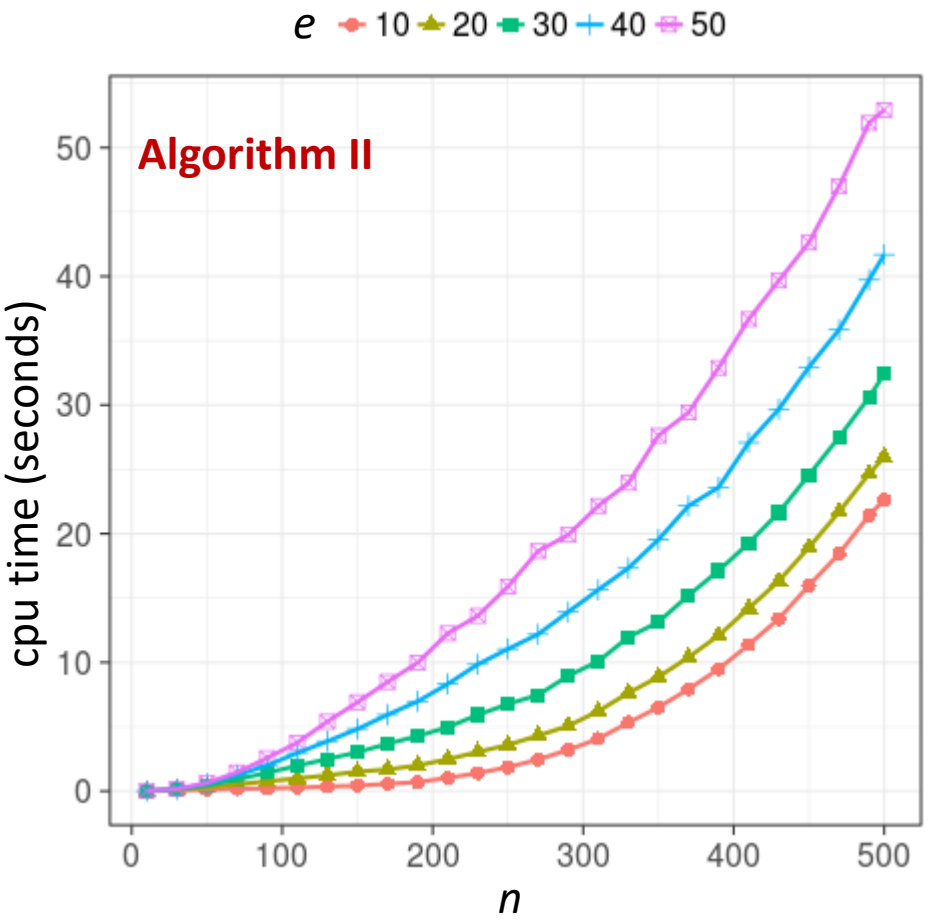


$|V_b| = O(k_i d_i + 1)$
$|E_b| = O(k_i d_i + 1)$

d_i = maximum in-degree in N_i
K_i = level of N_i

Implementation and Experiments





Source code: https://github.com/kmampent/ntd

Model

Build a random binary tree and add *e* random edges from an ancestor to a descendant

Publications

- Gerth Stølting Brodal and Konstantinos Mampentzidis.
  Cache Oblivious Algorithms for Computing the Triplet Distance between Trees.
  In *ESA 2017*, Vienna, Austria.

- Jesper Jansson, Konstantinos Mampentzidis, Ramesh Rajaby, and Wing-Kin Sung.
  Computing the Rooted Triplet Distance Between Phylogenetic Networks.
  In *IWOCA 2019*, Pisa, Italy.

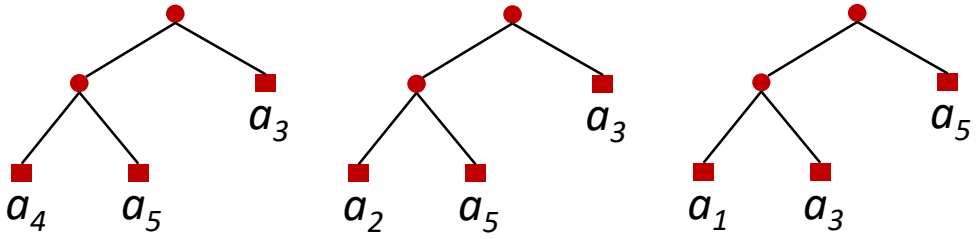
- **Jesper Jansson, Konstantinos Mampentzidis, and Sandhya Thekkumpadan Puthiyaveedu.**
  Building a Small and Informative Phylogenetic Supertree.
  In *WABI 2019*, Niagara Falls, USA.

Phylogenetic Supertrees

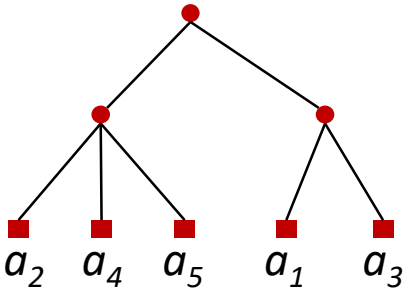
- **The Supertree Problem** Given a set R of small, accurate trees over overlapping subsets of n species, build a tree T that represents R as much as possible
- The output tree T is called a **phylogenetic supertree**

Example

R = set of rooted binary trees with three leaves



T = a rooted tree, *if it exists*, that has all trees from R as embedded subtrees



q-MAXRTC (q - Maximum Rooted Triplets Consistency)
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree with ***q* internal nodes** over Λ inducing the max # triplets from R

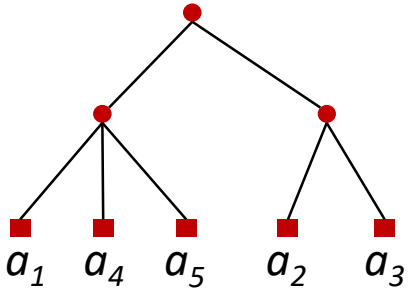
Example

$\Lambda = \{a_1, a_2, a_3, a_4, a_5\}$

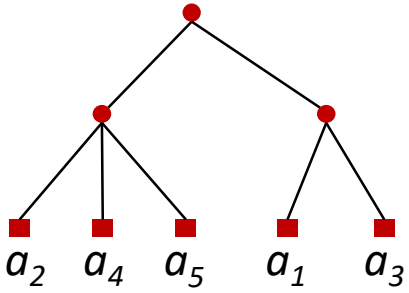
$n = 5$

$R = \{a_4a_5|a_3,\ a_2a_5|a_3,\ a_1a_3|a_5,\ a_2a_4|a_5,\ a_2a_3|a_1\}$

$q = 3$



value = 2



value = 3
optimal

Motivation – Related Work

q-MAXRTC (q - Maximum Rooted Triplets Consistency)
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree with **q internal nodes** over Λ inducing the max # triplets from R

Aho *et al.* [SICOMP 1981]
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree, *if it exists*, over Λ inducing all triplets from R

Solvable in polynomial time by the BUILD algorithm

- BUILD does not always return a tree with the min # internal nodes
- Jansson *et al.* [SICOMP 2012]: BUILD can return a tree with $\Omega(n)$ unnecessary internal nodes
  ⇒ may suggest false groupings of the leaves, also known as *spurious novel clades*
- Scientists typically look for simple explanations for a set of observations

MINRS (Minimally Resolved Supertree), Jansson *et al.* [SICOMP 2012]
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree, *if it exists*, with the min # internal nodes over Λ inducing all triplets from R

- The decision version of MINRS is NP-Hard when # internal nodes is ≥ 4, polynomial time solvable otherwise
- Very sensitive to outliers

Motivation – Related Work

q-MAXRTC (q - Maximum Rooted Triplets Consistency)
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree with **q internal nodes** over Λ inducing the max # triplets from R

MINRS (Minimally Resolved Supertree), Jansson *et al.* [SICOMP 2012]
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree, *if it exists*, with the min # internal nodes over Λ inducing all triplets from R

MAXRTC (Maximum Rooted Triplets Consistency), Bryant [PhD Thesis 1997]
R = set of resolved triplets over a leaf label set Λ of size n
T = rooted tree over Λ inducing the max # triplets from R

- MAXRTC is NP-Hard
- Polynomial-time approximation algorithms building trees that induce $\geq 1/3|R|$ triplets exist

Reference	Approximation ratio	T	# internal nodes
Gąsieniec *et al.* [JCO 1999]	1/3	caterpillar	unbounded
Byrka *et al.* [Discr. Appl. Math. 2010]	1/3	binary	n-1
Byrka *et al.* [JDA 2010]	1/3	binary	n-1

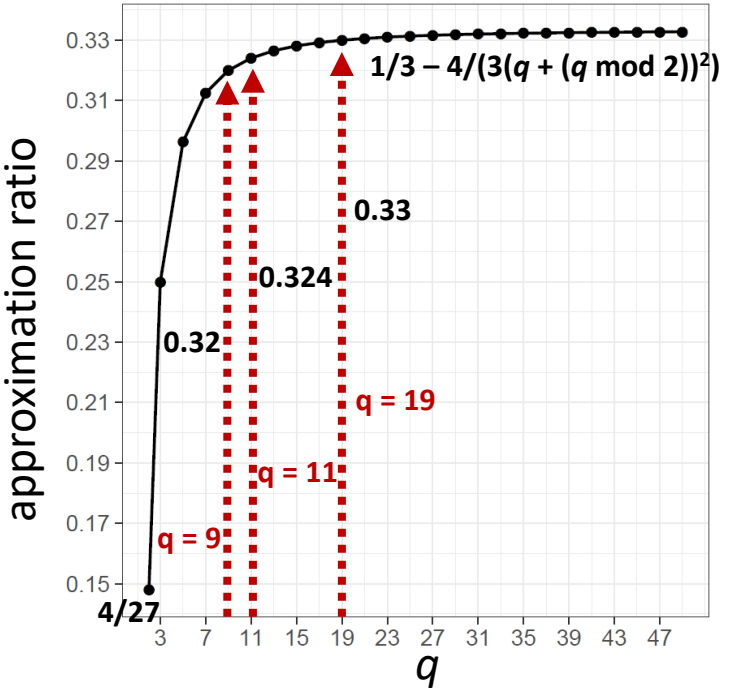
q-MAXRTC = MINRS + MAXRTC

Approximation Algorithms for *q*-MAXRTC

Reference	Deterministic	*q*	Approximation Ratio	Type
Gąsieniec *et al.* [JCO 1999]	yes	unbounded	1/3	abs.
Byrka *et al.* [Discr. Appl. Math. 2010]	yes	*n*-1	1/3	abs.
Byrka *et al.* [JDA 2010]	yes	*n*-1	1/3	abs.
new [WABI 2019]	no	2	1/2	rel.
new [WABI 2019]	yes	2	1/4	rel.
new [WABI 2019] ★	yes	2	4/27	abs.
new [WABI 2019] ★	yes	≥ 3	$1/3 - 4/(3(q + (q \bmod 2))^2)$	abs.

★ **Implementation available**

- n = size of the input leaf label set
- q = # internal nodes in output tree T
- Absolute approximation ratio r (abs.):
  T induces ≥ $r \cdot |R|$ triplets
- Relative approximation ratio r (rel.):
  T induces ≥ $r \cdot OPT$ triplets
  OPT = value of the optimal solution



Approximation Algorithms for q-MAXRTC

Reference	Deterministic	q	Approx. Ratio	Type
new [WABI 2019]	no	2	1/2	rel.
new [WABI 2019]	yes	2	1/4	rel.
new [WABI 2019]	yes	2	4/27	abs.
new [WABI 2019]	yes	≥ 3	$1/3 - 4/(3(q + (q \bmod 2))^2)$	abs.

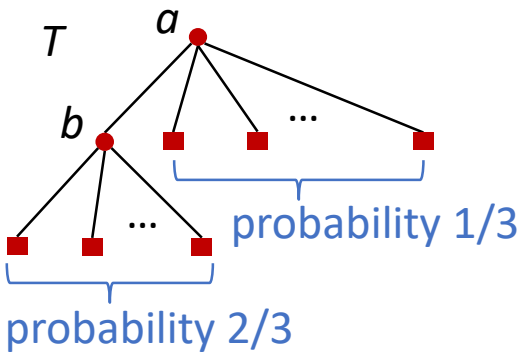
- Intuitively, the larger the value of q, the better must be the quality of the produced trees

Lemma 4

Let $2 \le q' \le q \le n-1$. We have that $opt(q') \le opt(q) \le \left\lceil \frac{q-1}{q'-1} \right\rceil opt(q')$

$q = 2$

1. Build a tree with two internal nodes labelled a and b
2. For each leaf: with probability 2/3 assign it to be the child of b, and with probability 1/3 the child of a



Expected # triplets consistent with T: $4|R|/27$

- The algorithm is derandomized in $O(|R|)$ time with the *method of conditional expectations*
- **Theorem 8**: 4/27 is the best possible absolute ratio

Approximation Algorithms for q-MAXRTC

Reference	Deterministic	q	Approx. Ratio	Type
new [WABI 2019]	no	2	1/2	rel.
new [WABI 2019]	yes	2	1/4	rel.
new [WABI 2019]	yes	2	4/27	abs.
new [WABI 2019]	yes	≥ 3	$1/3 - 4/(3(q + (q \bmod 2))^2)$	abs.

$q \geq 3$

First case: $q = 2k+1$ for some $k \in \mathbb{N}$

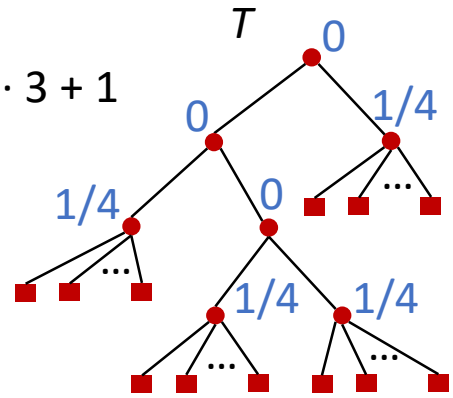
1. Build a binary tree with q nodes
2. Assignment probability for a node with children: 0
3. Assignment probability for a node without children: $1/(k+1)$
4. Assign all n leaves one by one

Expected # triplets consistent with T: $1/3 - 4/(3(q + 1)^2)$

example

$q = 7 = 2 \cdot 3 + 1$

$k = 3$

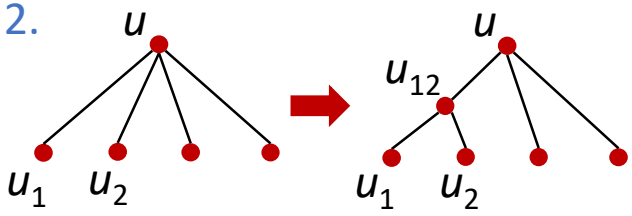


Second case: $q = 2k$ for some $k \in \mathbb{N}$

1. Apply first case for $q = q - 1$ and assign all n leaves
2. Add an extra internal node in T without reducing the total # of triplets induced by T from R

Expected # triplets consistent with T: $1/3 - 4/(3q^2)$

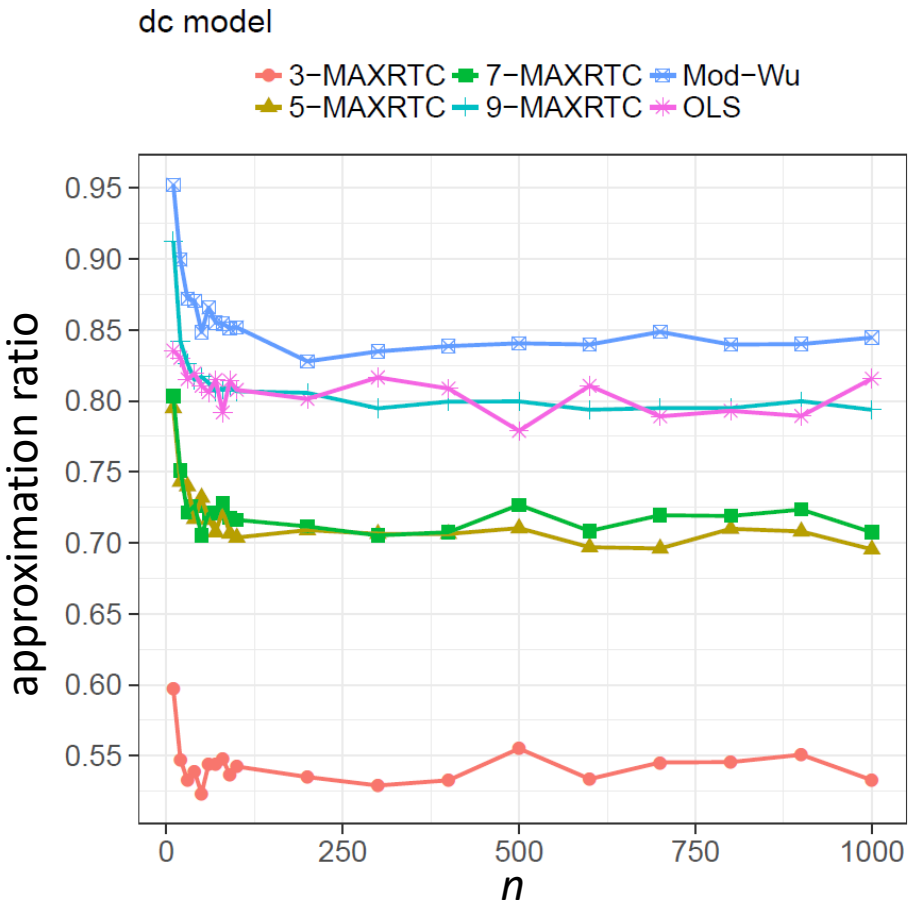
2.

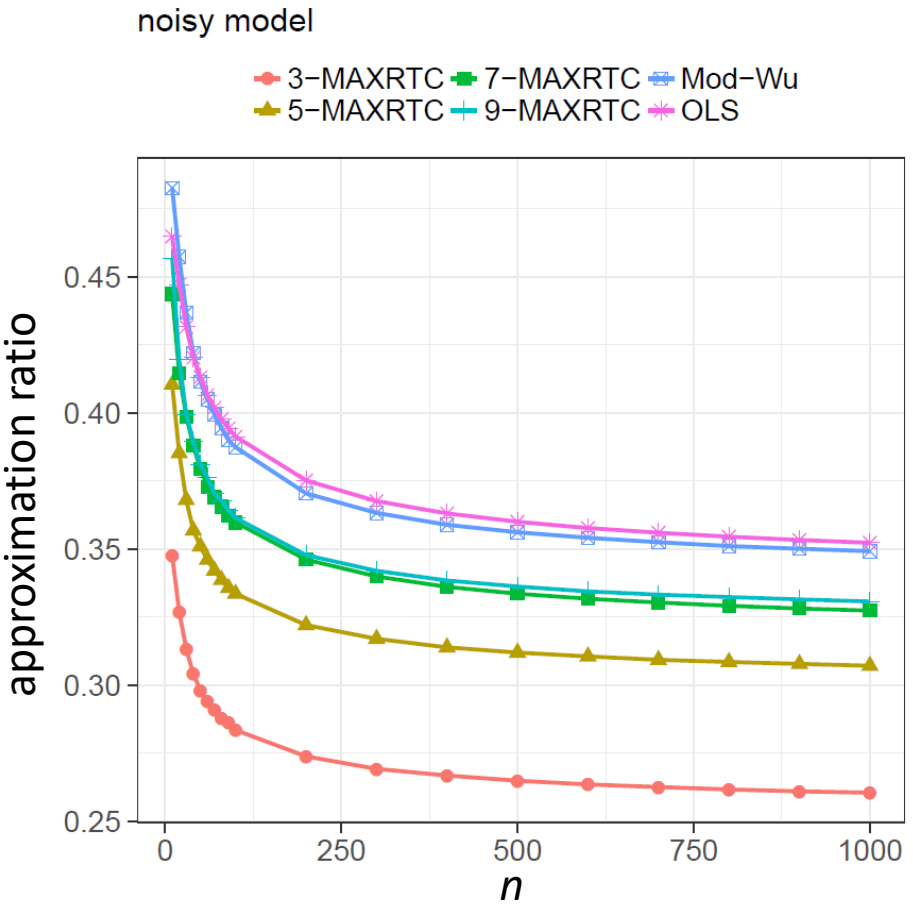


- The algorithm is derandomized in $O(q|R|)$ time with the *method of conditional expectations*
- **Open problem:** best possible absolute ratio?

q-MAXRTC – Implementation and Experiments

Experiments on Simulated Datasets





- *dc model*: *R* is defined by all the triplets extracted from a binary tree with *n* leaves
- *noisy model*: *R* contains random triplets

Source code: https://github.com/kmampent/qMAXRTC

q-MAXRTC – Implementation and Experiments

Experiments on Real Datasets

- Use five published **binary trees** from the following two papers:

  L. A. Hug *et al*. A new view of the tree of life. *Nature Microbiology*, 1, 2016.

  J. M. Lang *et al*. Phylogeny of bacterial and archaeal genomes using conserved genes: supertrees and supermatrices. *PLoS ONE*, 8(4), 2013.

- For every tree, extract n^2 triplets at random and use them to define R

q	poS1(761)	poS2(761)	poS4(841)	nmS4(1869)	nmS2(3082)	Average
2	0.27	0.36	0.43	0.41	0.29	0.35
3	0.67	0.54	0.48	0.41	0.46	0.51
5	0.77	0.81	0.67	0.66	0.72	0.73
7	0.82	0.75	0.76	0.62	0.73	0.74
9	0.86	0.71	0.87	0.80	0.79	0.81
11	0.91	0.89	0.87	0.79	0.87	0.87

ratio = $S(T_1, T_2)/\binom{n}{3}$, where $S(T_1, T_2)$ = # triplets that are induced by both T_1 and T_2 and n is inside the parenthesis

- With only 9 internal nodes we can capture on average 80% of the triplets

Source code: https://github.com/kmampent/qMAXRTC

q-MAXRTC – Implementation and Experiments

Experiments on Real Datasets

- Use five published **binary trees** from the following two papers:

  L. A. Hug *et al*. A new view of the tree of life. *Nature Microbiology*, 1, 2016.

  J. M. Lang *et al*. Phylogeny of bacterial and archaeal genomes using conserved genes: supertrees and supermatrices. *PLoS ONE*, 8(4), 2013.

- For every tree, extract n^2 triplets at random and use them to define R

q	poS1(761)	poS2(761)	poS4(841)	nmS4(1869)	nmS2(3082)
2	0.06	0.06	0.07	0.40	1.16
3	0.32	0.32	0.39	1.96	5.38
5	0.47	0.47	0.58	2.92	7.85
7	0.63	0.62	0.76	3.83	10.53
9	0.78	0.79	0.96	4.85	13.15
11	0.94	0.94	1.14	5.71	15.56

Running time in seconds

Source code: https://github.com/kmampent/qMAXRTC

Revisiting the Rooted Triplet Distance (Trees)

Reference	Time	Space	Non-Binary Trees
Critchlow *et al.* [Sys. Biology 1996]	$O(n^2)$	$O(n^2)$	no
Bansal *et al.* [TCS 2011]	$O(n^2)$	$O(n^2)$	yes
Sand *et al.* [BMC Bioinform. 2013] ★	$O(n \cdot \log^2 n)$	$O(n)$	no
Brodal *et al.* [SODA 2013] ★	$O(n \cdot \log n)$	$O(n \cdot \log n)$	yes
Jansson & Rajaby [JCB 2017] ★	$O(n \cdot \log^3 n)$	$O(n \cdot \log n)$	yes
Brodal & Mampentzidis [ESA 2017] ★	$O(n \cdot \log n)$	$O(n)$	yes
new [WABI 2019] ★	$O(q \cdot n)$	$O(q \cdot n)$	yes

★ **Implementation available**

- n = size of the common leaf label set between the two input trees
- q = # internal nodes in the smaller input tree

Open Problems

- $O(n \log n / \log\log n)$? $O(n)$?
- If q_1 is the total # internal nodes in T_1 and similarly q_2 in T_2, $O(q_1 q_2 + n)$?
- Prove any non-trivial lower bound

Summary

Rooted Triplet Distance (Trees) https://github.com/kmampent/{CacheTD,qtd}

Reference	Time	I/Os	Space	Non-Binary Trees
new [ESA 2017]	$O(n \cdot \log n)$	$O(n/B \cdot \log_2(n/M))$	$O(n)$	yes
new [WABI 2019]	$O(q \cdot n)$	$O(q \cdot n)$	$O(q \cdot n)$	yes

Rooted Triplet Distance (Networks) https://github.com/kmampent/ntd

Reference	k (level)	Degrees	Time
new [IWOCA 2019]	arbitrary	arbitrary	$O(N^2M + n^3)$
new [IWOCA 2019]	arbitrary	arbitrary	$O(M + k^3d^3n + n^3)$

***q*-MAXRTC** https://github.com/kmampent/qMAXRTC

Reference	Deterministic	q	Approximation Ratio	Type
new [WABI 2019]	no	2	1/2	rel.
new [WABI 2019]	yes	2	1/4	rel.
new [WABI 2019]	yes	2	4/27	abs.
new [WABI 2019]	yes	≥ 3	$1/3 - 4/(3(q + (q \bmod 2))^2)$	abs.