The Approximability of Maximum Rooted Triplets Consistency with Fan Triplets and Forbidden Triplets

- Jesper Jansson
  (Kyoto University, Japan)
- Andrzej Lingas
  (Lund University, Sweden)
- Eva-Marta Lundell
  (Lund University, Sweden)

Introduction

Definition

A phylogenetic tree is a rooted, unordered tree whose leaves are uniquely labeled and in which every internal node has ≥ 2 children.

Introduction

Definition

A phylogenetic tree is a rooted, unordered tree whose leaves are uniquely labeled and in which every internal node has ≥ 2 children.

Can describe divergent evolutionary history for a set of objects, where:

"objects" = Biological species, categories of species, populations, proteins, nucleic acids, natural languages, chain letters, medieval manuscripts, or ...

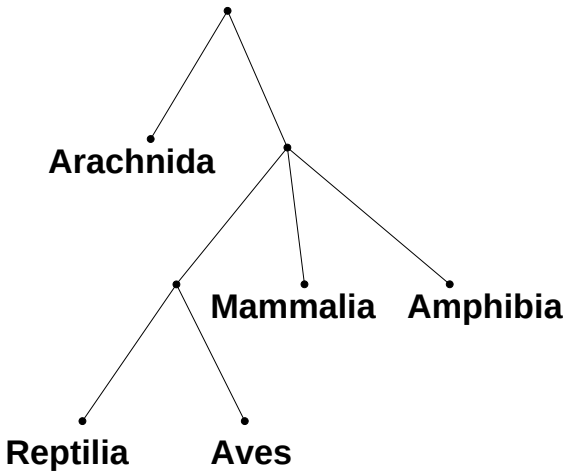
Introduction

Definition

A phylogenetic tree is a rooted, unordered tree whose leaves are uniquely labeled and in which every internal node has ≥ 2 children.

Can describe divergent evolutionary history for a set of objects, where:

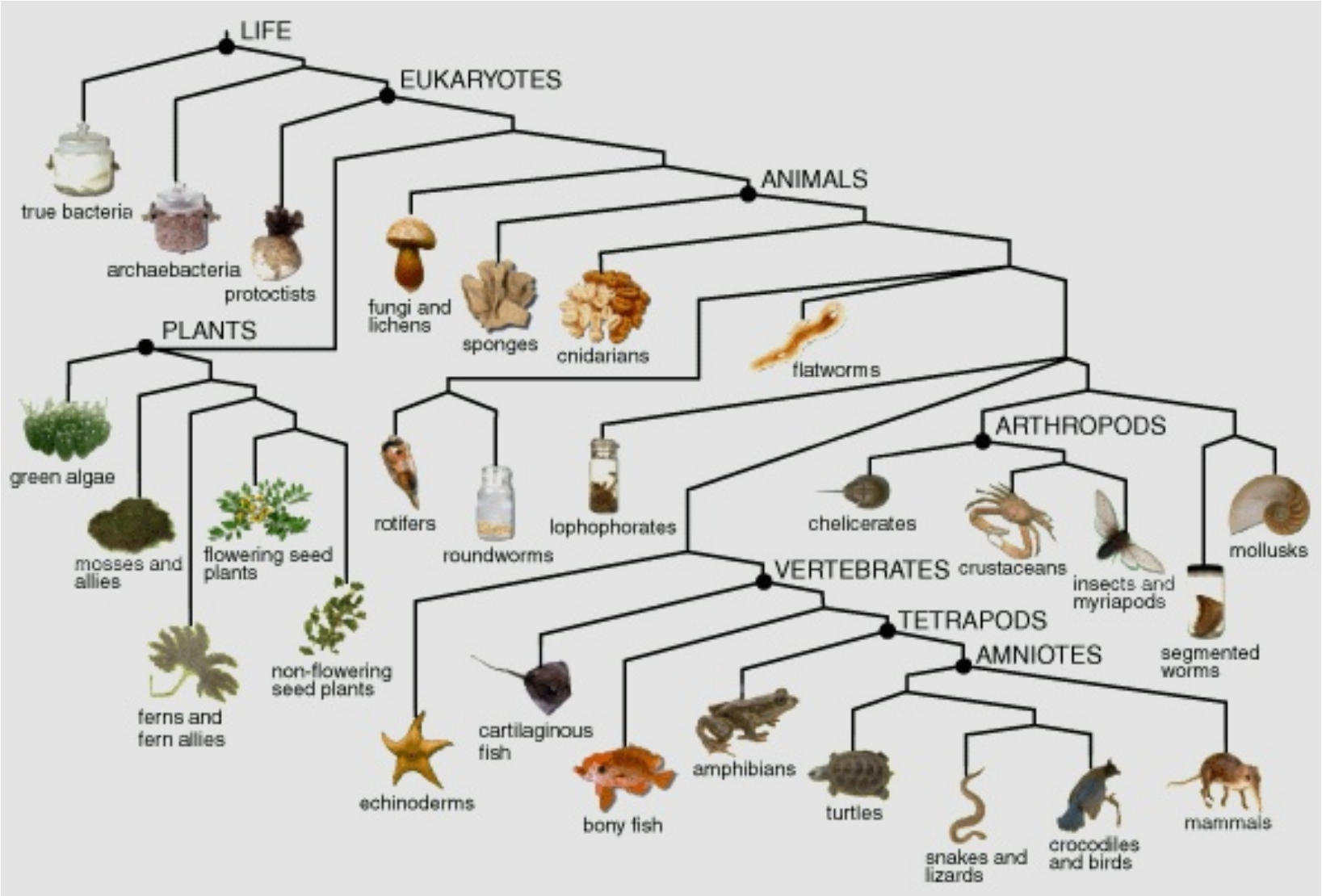
"objects" = Biological species, categories of species, populations, proteins, nucleic acids, natural languages, chain letters, medieval manuscripts, or ...



Main idea:

- Represent objects by *leaves* in the tree.
- Select branching structure so that *internal nodes* correspond to common ancestors.

Example



(Figure from `http://biology.unm.edu/ccouncil/Biology_203`)

Inferring phylogenetic trees

Important problem:

How can we build a phylogenetic tree?

Inferring phylogenetic trees

Important problem:
How can we build a phylogenetic tree?

(Not a new problem!)

Inferring phylogenetic trees

Important problem:
How can we build a phylogenetic tree?

(Not a new problem!)

- During the last 150 years, numerous methods for reconstructing phylogenetic trees have been proposed.

Inferring phylogenetic trees

Important problem:
How can we build a phylogenetic tree?

(Not a new problem!)

- During the last 150 years, numerous methods for reconstructing phylogenetic trees have been proposed.
- For various reasons, inferring an accurate phylogenetic tree can be a difficult problem.

Inferring phylogenetic trees

Important problem:
How can we build a phylogenetic tree?

(Not a new problem!)

- During the last 150 years, numerous methods for reconstructing phylogenetic trees have been proposed.
- For various reasons, inferring an accurate phylogenetic tree can be a difficult problem.
- For example, small changes in the input data may produce trees with very different structures.

Inferring phylogenetic trees

Important problem:
How can we build a phylogenetic tree?

(Not a new problem!)

- During the last 150 years, numerous methods for reconstructing phylogenetic trees have been proposed.
- For various reasons, inferring an accurate phylogenetic tree can be a difficult problem.
- For example, small changes in the input data may produce trees with very different structures.
- Furthermore, many of the underlying computational problems are $\mathcal{NP}$-hard optimization problems.

Inferring phylogenetic trees, cont.

Different types of data available. $\Rightarrow$ Different methods may be appropriate.

This talk is focused on one particular method:
The phylogenetic supertree approach

Phylogenetic supertree

Goal:

Merge a given set of (possibly conflicting) phylogenetic trees with overlapping leaf label sets into *one tree*. This is called a supertree. Keep as much branching information as possible!

Phylogenetic supertree

Goal:

Merge a given set of (possibly conflicting) phylogenetic trees with overlapping leaf label sets into *one tree*. This is called a supertree. Keep as much branching information as possible!

Motivation:

- Combine many trees constructed from different data sets.
  $\Rightarrow$ More reliable answers.

Phylogenetic supertree

Goal:

Merge a given set of (possibly conflicting) phylogenetic trees with overlapping leaf label sets into *one tree*. This is called a supertree. Keep as much branching information as possible!

Motivation:

- Combine many trees constructed from different data sets.
  $\Rightarrow$ More reliable answers.
- Most individual studies investigate relatively few species. Supertrees allow us to deduce new, hypothetical evolutionary relationships.

Phylogenetic supertree

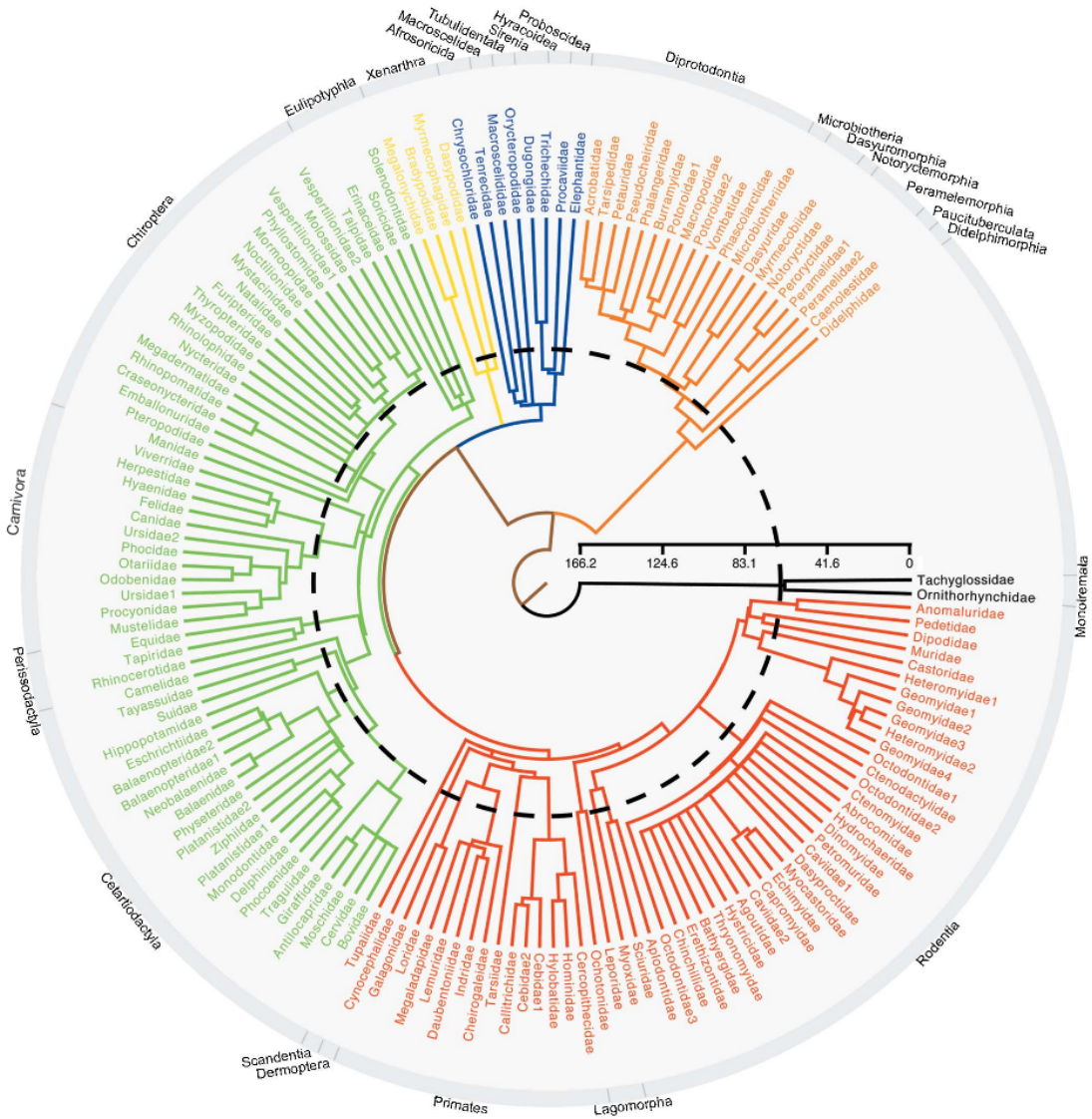
Goal:

Merge a given set of (possibly conflicting) phylogenetic trees with overlapping leaf label sets into *one tree*. This is called a supertree. Keep as much branching information as possible!

Motivation:

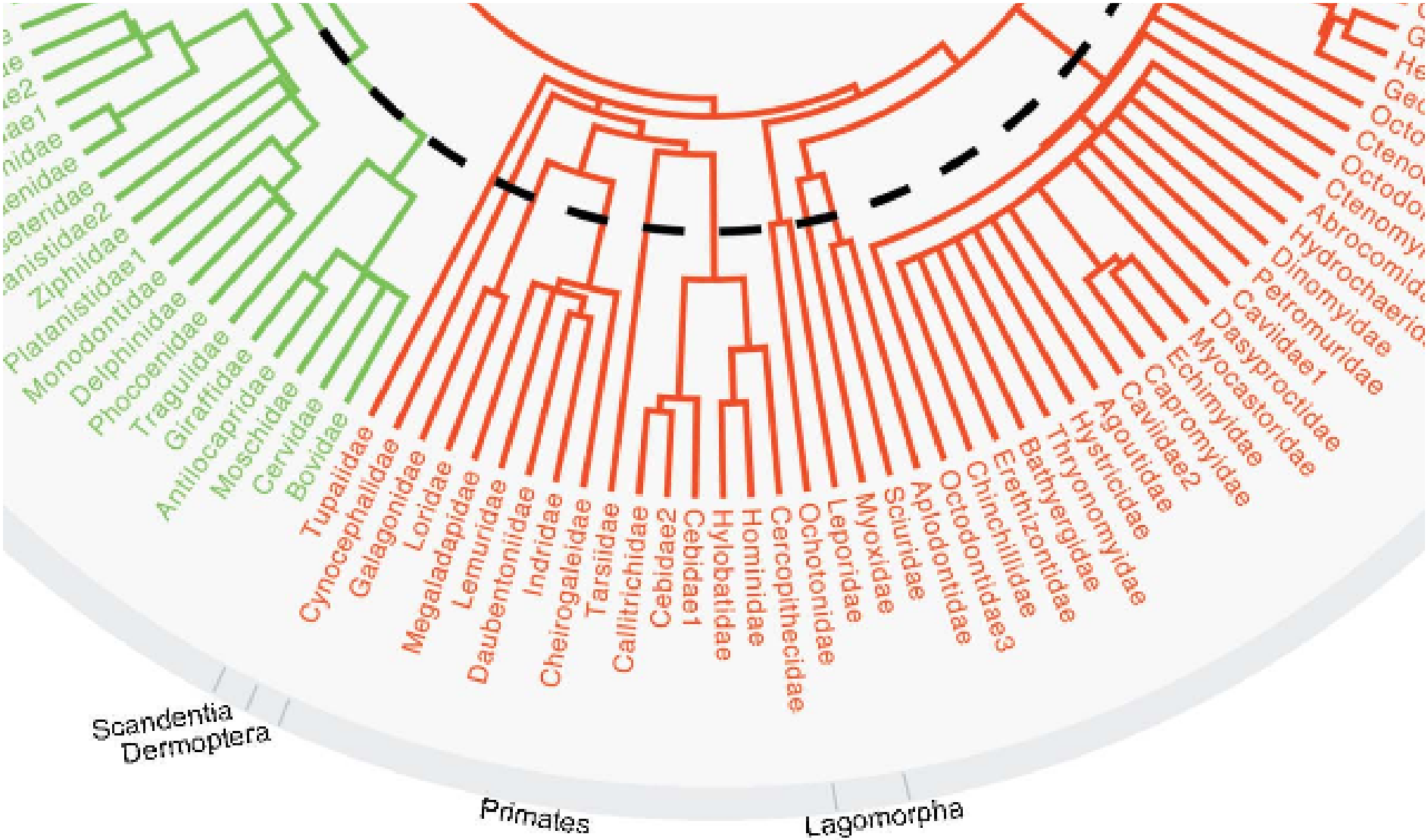
- Combine many trees constructed from different data sets.
  $\Rightarrow$ More reliable answers.
- Most individual studies investigate relatively few species.
  Supertrees allow us to deduce new, hypothetical evolutionary relationships.
- Computationally expensive methods can yield highly accurate phylogenetic trees for small, overlapping subsets of the objects.
  $\Rightarrow$ A "divide-and-conquer"-based technique for inferring large trees.
  Has become popular in recent years.

Phylogenetic supertree, example



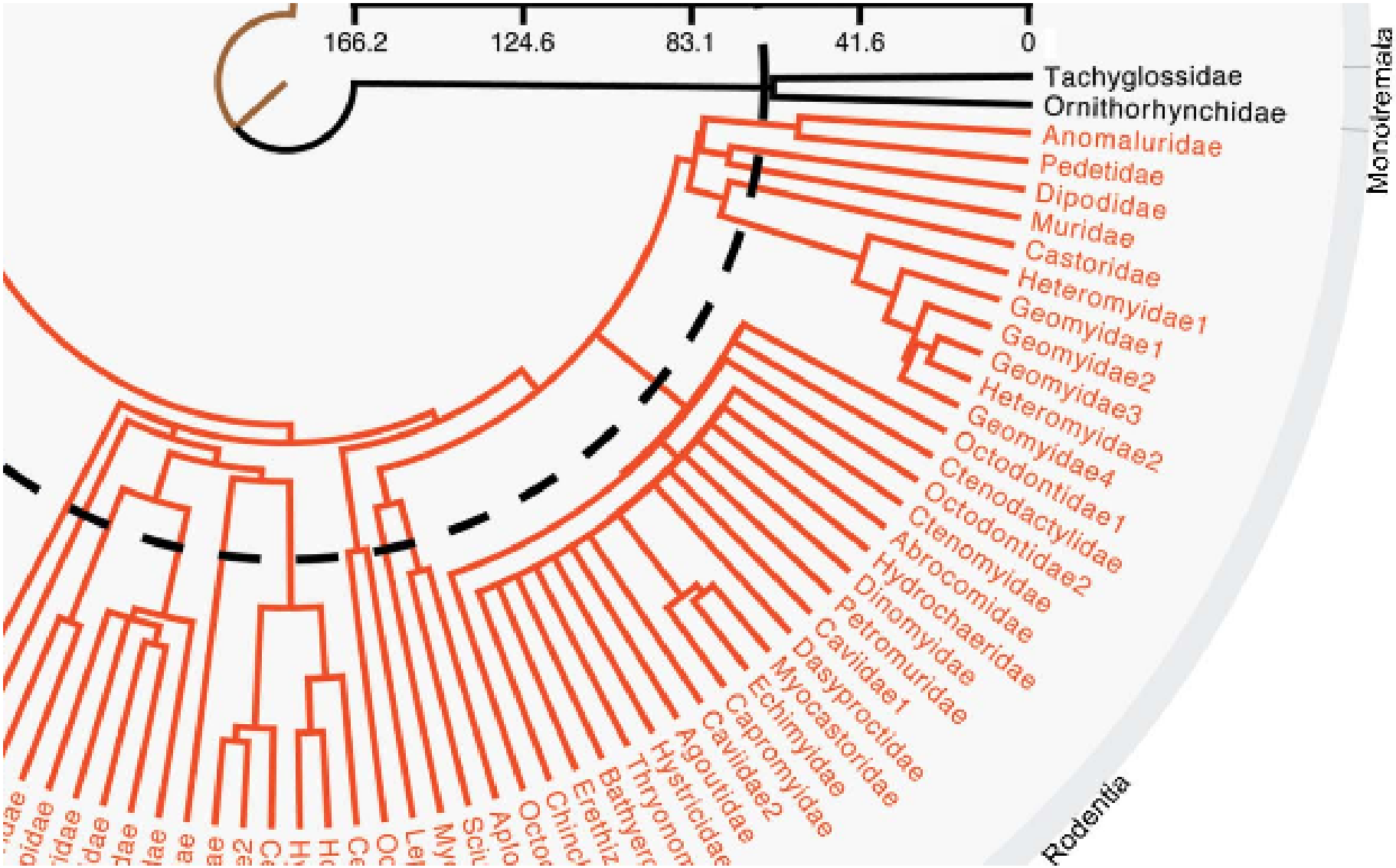
(From O. R. P. Bininda-Emonds *et al.*: "The delayed rise of present-day mammals", *Nature*, Vol. 446, pp. 507–512, 2007).

Phylogenetic supertree, example



(From O. R. P. Bininda-Emonds *et al.*: "The delayed rise of present-day mammals", *Nature*, Vol. 446, pp. 507–512, 2007).

Phylogenetic supertree, example



(From O. R. P. Bininda-Emonds *et al.*: "The delayed rise of present-day mammals", *Nature*, Vol. 446, pp. 507–512, 2007).

- From here on, "tree" = "phylogenetic tree" (i.e., rooted, unordered tree with uniquely labeled leaves in which every internal node has ≥ 2 children.).

- From here on, "tree" = "phylogenetic tree" (i.e., rooted, unordered tree with uniquely labeled leaves in which every internal node has ≥ 2 children.).
- Also, every leaf is identified with its label.

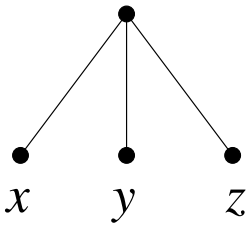
Notation

A tree with exactly three leaves is called a rooted triplet.

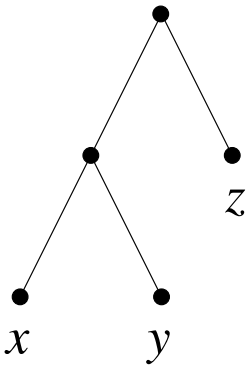
Notation

A tree with exactly three leaves is called a rooted triplet.

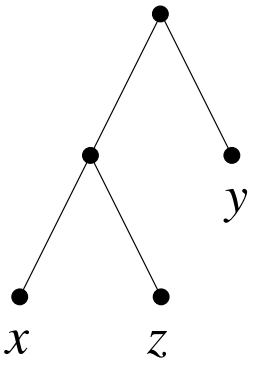
Let $\{x, y, z\}$ be a leaf label set of cardinality 3.
There are four different possible trees leaf-labeled by $\{x, y, z\}$:



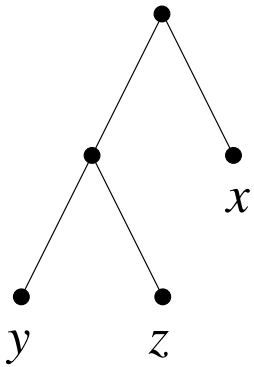
$x \mid y \mid z$



$x\ y \mid z$



$x\ z \mid y$

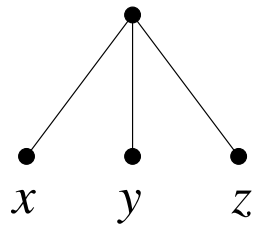


$y\ z \mid x$

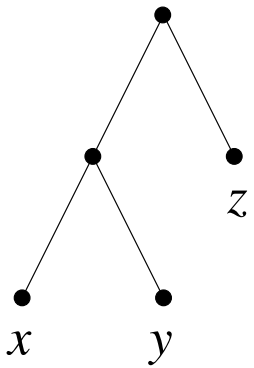
Notation

A tree with exactly three leaves is called a rooted triplet.

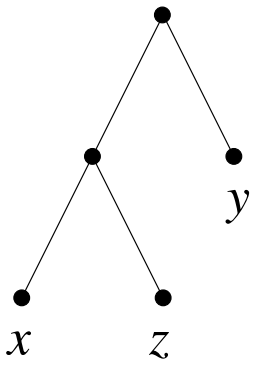
Let $\{x, y, z\}$ be a leaf label set of cardinality 3.
There are four different possible trees leaf-labeled by $\{x, y, z\}$:



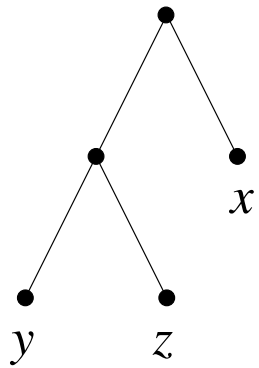
$x \mid y \mid z$



$x\ y \mid z$



$x\ z \mid y$



$y\ z \mid x$

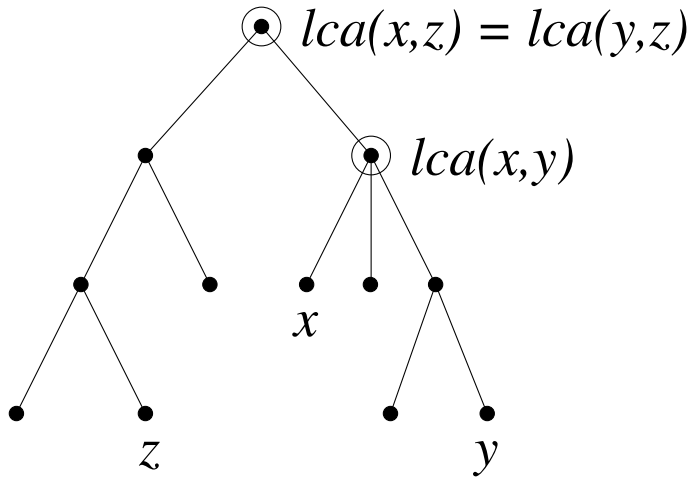
Two types of rooted triplets:

- *Fan triplet* = One internal node $(x|y|z)$
- *Resolved triplet* = Two internal nodes $(xy|z,\ xz|y,$ and $yz|x)$

Notation, cont.

Let x, y, z be three leaves in a tree T.

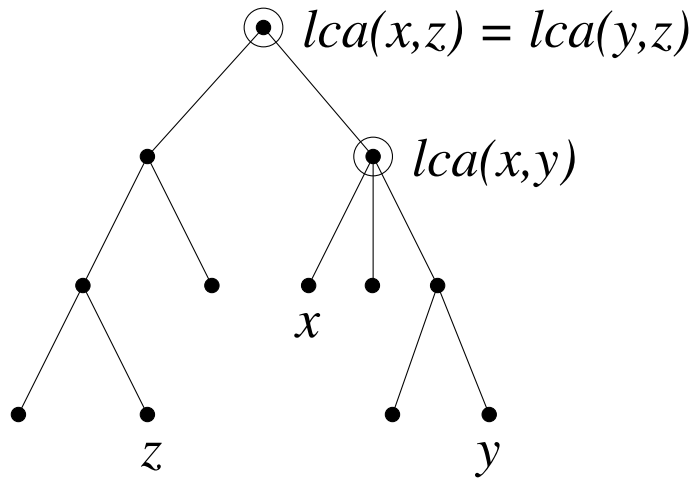
- If $lca(x, y)$ is a proper descendant of $lca(x, z) = lca(y, z)$ in T then we say that T and the resolved triplet $xy|z$ are *consistent*.



Notation, cont.

Let x, y, z be three leaves in a tree T.

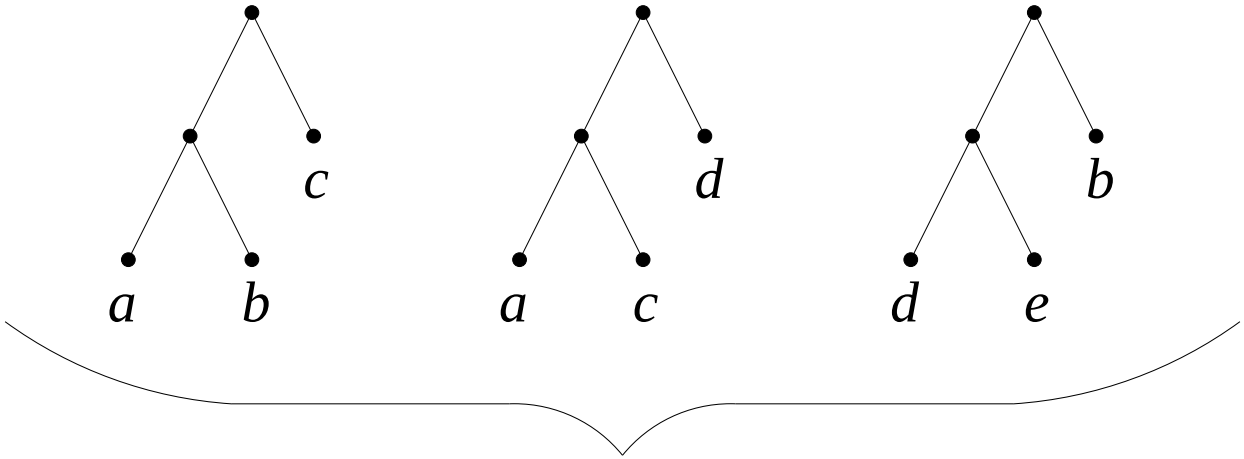
- If $lca(x, y)$ is a proper descendant of $lca(x, z) = lca(y, z)$ in T then we say that T and the resolved triplet $xy|z$ are *consistent*.



- On the other hand, if $lca(x, y) = lca(x, z) = lca(y, z)$ in T then T and the fan triplet $x|y|z$ are *consistent*.

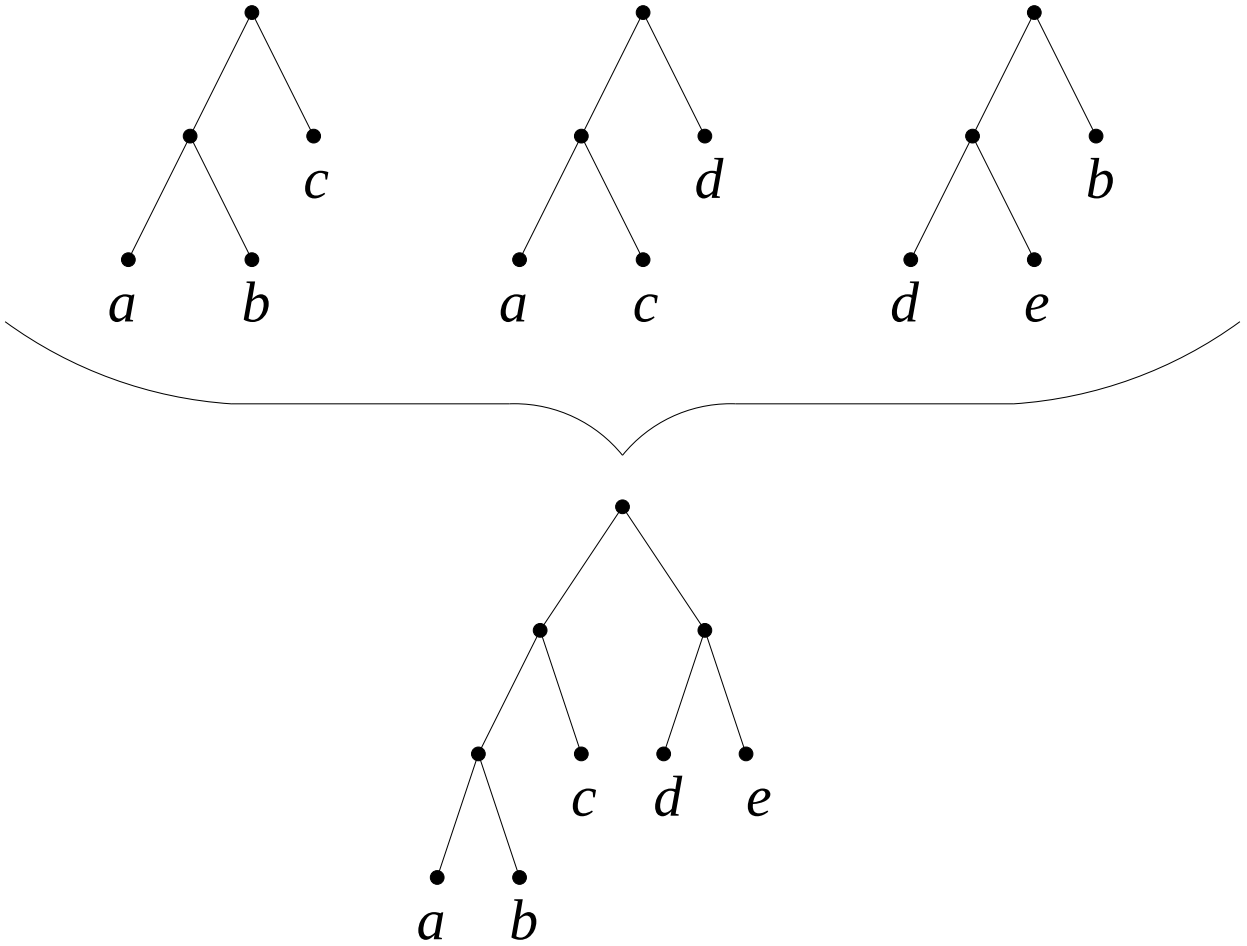
Combining a set of rooted triplets

Example:



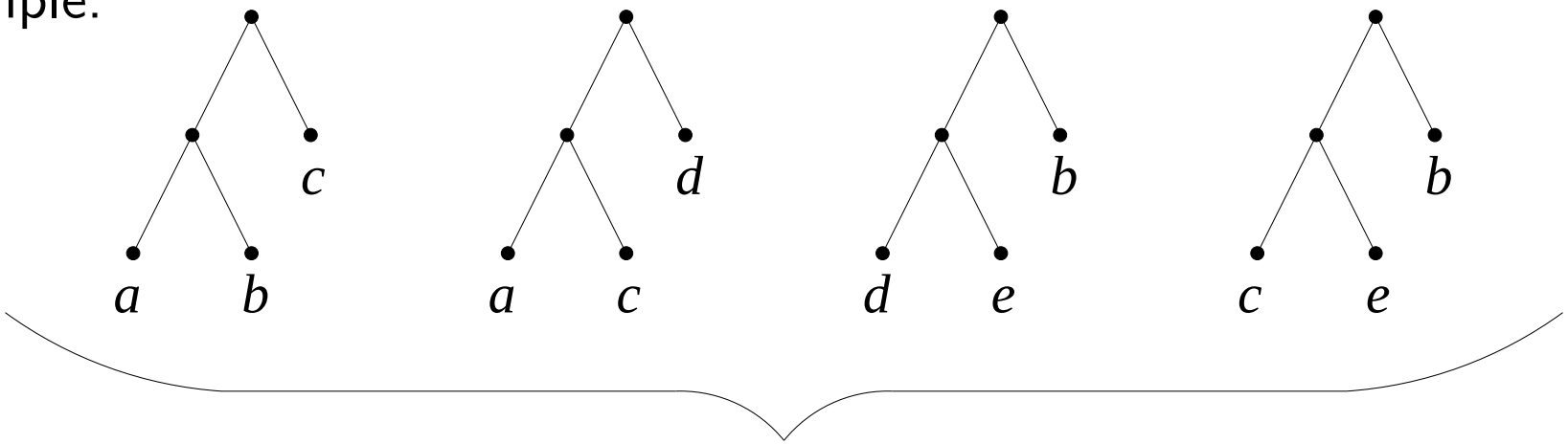
Combining a set of rooted triplets

Example:



Combining a set of rooted triplets

Example:



CONFLICT!!!

Problem definitions

Notation:

- For any tree T, define $\Lambda(T) =$ the set of all leaf labels in T.

Problem definitions

Notation:

- For any tree T, define $\Lambda(T) =$ the set of all leaf labels in T.
- For any tree T and any set $\mathcal{X}$ of rooted triplets with $\bigcup_{t \in \mathcal{X}} \Lambda(t) \subseteq \Lambda(T)$, define $T(\mathcal{X}) = \{t \in \mathcal{X} : t \text{ is consistent with } T\}$.

Problem definitions

Notation:

- For any tree T, define $\Lambda(T) =$ the set of all leaf labels in T.
- For any tree T and any set $\mathcal{X}$ of rooted triplets with $\bigcup_{t \in \mathcal{X}} \Lambda(t) \subseteq \Lambda(T)$, define $T(\mathcal{X}) = \{t \in \mathcal{X} : t \text{ is consistent with } T\}$.

The Maximum Rooted Triplets Consistency Problem (MTC)

Input: Two sets $\mathcal{G}$ and $\mathcal{B}$ of rooted triplets.

Output: Tree T with $\Lambda(T) = \bigcup_{t \in \mathcal{G} \cup \mathcal{B}} \Lambda(t)$ maximizing $\boxed{|T(\mathcal{G})| + |\mathcal{B} \setminus T(\mathcal{B})|}$.

Problem definitions

Notation:

- For any tree T, define $\Lambda(T) =$ the set of all leaf labels in T.
- For any tree T and any set $\mathcal{X}$ of rooted triplets with $\bigcup_{t \in \mathcal{X}} \Lambda(t) \subseteq \Lambda(T)$, define $T(\mathcal{X}) = \{t \in \mathcal{X} : t \text{ is consistent with } T\}$.

The Maximum Rooted Triplets Consistency Problem (MTC)

Input: Two sets $\mathcal{G}$ and $\mathcal{B}$ of rooted triplets.

Output: Tree T with $\Lambda(T) = \bigcup_{t \in \mathcal{G} \cup \mathcal{B}} \Lambda(t)$ maximizing $|T(\mathcal{G})| + |\mathcal{B} \setminus T(\mathcal{B})|$.

The input rooted triplets can be interpreted as *constraints*:

- Resolved triplets in $\mathcal{G}$ = "constraints of type 1"
- Fan triplets in $\mathcal{G}$ = "constraints of type 2"
- Resolved triplets in $\mathcal{B}$ = "constraints of type 3" = "forbidden resolved triplets"
- Fan triplets in $\mathcal{B}$ = "constraints of type 4" = "forbidden fan triplets"

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.
- He, Huynh, Jansson, Sung (2006):
  Extension of the above to the case of forbidden resolved triplets.

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.
- He, Huynh, Jansson, Sung (2006):
  Extension of the above to the case of forbidden resolved triplets.
- Bryant (1997):
  MTC (i.e., the optimization version of rooted triplets consistency) is NP-hard, even if restricted to resolved triplets.

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.
- He, Huynh, Jansson, Sung (2006):
  Extension of the above to the case of forbidden resolved triplets.
- Bryant (1997):
  MTC (i.e., the optimization version of rooted triplets consistency) is NP-hard, even if restricted to resolved triplets.
- Byrka, Gawrychowski, Huber, Kelk (2010):
  MTC is APX-hard, even if restricted to resolved triplets.

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.
- He, Huynh, Jansson, Sung (2006):
  Extension of the above to the case of forbidden resolved triplets.
- Bryant (1997):
  MTC (i.e., the optimization version of rooted triplets consistency) is NP-hard, even if restricted to resolved triplets.
- Byrka, Gawrychowski, Huber, Kelk (2010):
  MTC is APX-hard, even if restricted to resolved triplets.
- Gąsieniec, Jansson, Lingas, Östlin (1999):
  Polynomial-time $\frac{1}{3}$-approximation for MTC restricted to resolved triplets.

Previous work

- Aho, Sagiv, Szymanski, Ullman (1981):
  Polynomial-time algorithm to determine if there is a tree consistent with all of the resolved triplets in a set, and if so, to output such a tree.
- He, Huynh, Jansson, Sung (2006):
  Extension of the above to the case of forbidden resolved triplets.
- Bryant (1997):
  MTC (i.e., the optimization version of rooted triplets consistency) is NP-hard, even if restricted to resolved triplets.
- Byrka, Gawrychowski, Huber, Kelk (2010):
  MTC is APX-hard, even if restricted to resolved triplets.
- Gąsieniec, Jansson, Lingas, Östlin (1999):
  Polynomial-time $\frac{1}{3}$-approximation for MTC restricted to resolved triplets.
- Wu (2004):
  Exact $O(3^n(m+n^2))$-time algorithm for MTC restricted to resolved triplets, where n = number of leaf labels and m = number of input triplets.

Our new results

1. Polynomial-time $\frac{1}{4}$-approximation algorithm for MTC.

Our new results

1. Polynomial-time $\frac{1}{4}$-approximation algorithm for MTC.
2. Extension of Wu's exact exponential-time algorithm for MTC restricted to resolved triplets to also allow fan triplets, forbidden resolved triplets, and forbidden fan triplets.

   It takes an additional parameter $k \geq 2$ as input and forces the output to be a tree in which every internal node has at most k children.

   $\Rightarrow$ Running time: $\boxed{O((k+1)^{n+1}(m+n))}$

Our new results

1. Polynomial-time $\frac{1}{4}$-approximation algorithm for MTC.
2. Extension of Wu's exact exponential-time algorithm for MTC restricted to resolved triplets to also allow fan triplets, forbidden resolved triplets, and forbidden fan triplets.

   It takes an additional parameter $k \geq 2$ as input and forces the output to be a tree in which every internal node has at most k children.

   $\Rightarrow$ Running time: $\boxed{O((k+1)^{n+1}(m+n))}$
3. Exponential-time approximation scheme (ETAS) for MTC:

   For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in $\boxed{O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m+n))}$ time.

Our new results

1. Polynomial-time $\frac{1}{4}$-approximation algorithm for MTC.

2. Extension of Wu's exact exponential-time algorithm for MTC restricted to resolved triplets to also allow fan triplets, forbidden resolved triplets, and forbidden fan triplets.

   It takes an additional parameter $k \geq 2$ as input and forces the output to be a tree in which every internal node has at most k children.

   $\Rightarrow$ Running time: $\boxed{O((k+1)^{n+1}(m+n))}$

3. Exponential-time approximation scheme (ETAS) for MTC:

   For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in $\boxed{O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m+n))}$ time.

4. Polynomial-time approximation scheme (PTAS) for dense MTC, where an instance is called dense if it contains $\Omega(n^3)$ constraints:

   For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in polyn. time.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

3: **else**

Apply the polynomial-time $\frac{1}{3}$-approximation algorithm for MTC restricted to resolved triplets by [Gąsieniec *et al.* (1999)] to $\mathcal{R}_1$.

Output the resulting tree.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

3: **else**

Apply the polynomial-time $\frac{1}{3}$-approximation algorithm for MTC restricted to resolved triplets by [Gąsieniec *et al.* (1999)] to $\mathcal{R}_1$.

Output the resulting tree.

Theorem: Algorithm 1 is a polynomial-time $\frac{1}{4}$-approximation for MTC.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

3: **else**
Apply the polynomial-time $\frac{1}{3}$-approximation algorithm for MTC restricted to resolved triplets by [Gąsieniec *et al.* (1999)] to $\mathcal{R}_1$.
Output the resulting tree.

Theorem: Algorithm 1 is a polynomial-time $\frac{1}{4}$-approximation for MTC.

Proof: Let T be the output of the algorithm. There are two cases.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

3: **else**

Apply the polynomial-time $\frac{1}{3}$-approximation algorithm for MTC restricted to resolved triplets by [Gąsieniec *et al.* (1999)] to $\mathcal{R}_1$.

Output the resulting tree.

Theorem: Algorithm 1 is a polynomial-time $\frac{1}{4}$-approximation for MTC.

Proof: Let T be the output of the algorithm. There are two cases.

- $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$: Then T satisfies at least $\frac{m}{4}$ input constraints.

1. A 1/4-approximation algorithm for MTC

Algorithm 1

Input: Leaf label set S, set $\mathcal{R}$ of rooted triplet constraints over S.

1: Write $m = |\mathcal{R}|$. Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$ be the resolved triplets, fan triplets, forbidden resolved triplets, and forbidden fan triplets in $\mathcal{R}$, respectively.

2: **if** $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$ **then** output a tree consisting of a root node with n children labeled by S.

3: **else**
Apply the polynomial-time $\frac{1}{3}$-approximation algorithm for MTC restricted to resolved triplets by [Gąsieniec *et al.* (1999)] to $\mathcal{R}_1$.
Output the resulting tree.

Theorem: Algorithm 1 is a polynomial-time $\frac{1}{4}$-approximation for MTC.

Proof: Let T be the output of the algorithm. There are two cases.

- $|\mathcal{R}_2| + |\mathcal{R}_3| \geq \frac{m}{4}$: Then T satisfies at least $\frac{m}{4}$ input constraints.
- $|\mathcal{R}_1| + |\mathcal{R}_4| > \frac{3m}{4}$: The output of [Gąsieniec *et al.* (1999)] is a binary tree consistent with at least $\frac{1}{3}$ of the input resolved triplets.
  $\Rightarrow$ T satisfies $\geq \frac{1}{3}|\mathcal{R}_1| + |\mathcal{R}_4| \geq \frac{1}{3}(|\mathcal{R}_1| + |\mathcal{R}_4|) > \frac{1}{3} \cdot \frac{3m}{4} = \frac{m}{4}$ constraints. □

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.
- More precisely: For each such $U \subseteq S$, consider all bipartitions of U.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.
- More precisely: For each such $U \subseteq S$, consider all bipartitions of U.

  For each bipartition (U_1, U_2), count how many input resolved triplets are of the form $ab|c$, where $a \in U_i,\ b \in U_{3-i},\ c \notin U$. Call this $w(U_1, U_2)$.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.
- More precisely: For each such $U \subseteq S$, consider all bipartitions of U.

  For each bipartition (U_1, U_2), count how many input resolved triplets are of the form $ab|c$, where $a \in U_i,\ b \in U_{3-i},\ c \notin U$. Call this $w(U_1, U_2)$.

  Compute $score(U) = \max_{\text{bipartition } (U_1,U_2) \text{ of } U} \{score(U_1) + score(U_2) + w(U_1, U_2)\}$.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.
- More precisely: For each such $U \subseteq S$, consider all bipartitions of U.

  For each bipartition (U_1, U_2), count how many input resolved triplets are of the form $ab|c$, where $a \in U_i,\ b \in U_{3-i},\ c \notin U$. Call this $w(U_1, U_2)$.

  Compute $score(U) = \max_{\text{bipartition } (U_1,U_2) \text{ of } U} \{score(U_1) + score(U_2) + w(U_1, U_2)\}$.
- Traceback to recover an optimal tree.

2. An exact algorithm for MTC

Based on:

- Wu (2004):
  Exact exponential-time algorithm for MTC restricted to resolved triplets.

Wu's algorithm:

- Uses dynamic programming.
- Considers all subsets of the leaf label set S, in order of increasing cardinality.
- More precisely: For each such $U \subseteq S$, consider all bipartitions of U.

  For each bipartition (U_1, U_2), count how many input resolved triplets are of the form $ab|c$, where $a \in U_i,\ b \in U_{3-i},\ c \notin U$. Call this $w(U_1, U_2)$.

  Compute $score(U) = \max_{\text{bipartition } (U_1,U_2) \text{ of } U} \{score(U_1) + score(U_2) + w(U_1, U_2)\}$.
- Traceback to recover an optimal tree.
- $\Rightarrow O(3^n(m + n^2))$ time, where $n = |S|$ and $m =$ number of input triplets.

2. An exact algorithm for MTC, cont.

The new algorithm:

- **Notation:** For any $U \subseteq S$ and any partition P of U with $|P| \geq 2$, let
  - $w_2(P)$ = the number of input resolved triplets $ab|c$ such that a and b belong to two different parts in P and $c \notin U$.
  - $w_3(P)$ = the number of input fan triplets $a|b|c$ such that a, b, c belong to three different parts in P.
  - $w_{f2}(P)$ = the number of input forbidden resolved triplets $\neg(ab|c)$ such that a and c belong to two different parts in P and $b \notin U$, or b and c belong to two different parts in P and $a \notin U$, or a, b, c belong to three different parts in P.
  - $w_{f3}(P)$ = the number of input forbidden fan triplets $\neg(a|b|c)$ such that two elements in $\{a, b, c\}$ belong to two different parts in P and the remaining one does not belong to U.

2. An exact algorithm for MTC, cont.

The new algorithm:

- **Notation:** For any $U \subseteq S$ and any partition P of U with $|P| \geq 2$, let
  - $w_2(P)$ = the number of input resolved triplets $ab|c$ such that a and b belong to two different parts in P and $c \notin U$.
  - $w_3(P)$ = the number of input fan triplets $a|b|c$ such that a, b, c belong to three different parts in P.
  - $w_{f2}(P)$ = the number of input forbidden resolved triplets $\neg(ab|c)$ such that a and c belong to two different parts in P and $b \notin U$, or b and c belong to two different parts in P and $a \notin U$, or a, b, c belong to three different parts in P.
  - $w_{f3}(P)$ = the number of input forbidden fan triplets $\neg(a|b|c)$ such that two elements in $\{a, b, c\}$ belong to two different parts in P and the remaining one does not belong to U.
- Let $I(T)$ be the set of internal nodes in a tree T. For $v \in I(T)$, denote: T_v = the subtree of T rooted at v, $\{v_1, \ldots, v_l\}$ = the set of children of v, and π_v = the partition of $\Lambda(T_v)$ into $\Lambda(T_{v_1}), \ldots, \Lambda(T_{v_l})$.

2. An exact algorithm for MTC, cont.

The new algorithm:

- **Notation:** For any $U \subseteq S$ and any partition P of U with $|P| \geq 2$, let
  - $w_2(P)$ = the number of input resolved triplets $ab|c$ such that a and b belong to two different parts in P and $c \notin U$.
  - $w_3(P)$ = the number of input fan triplets $a|b|c$ such that a, b, c belong to three different parts in P.
  - $w_{f2}(P)$ = the number of input forbidden resolved triplets $\neg(ab|c)$ such that a and c belong to two different parts in P and $b \notin U$, or b and c belong to two different parts in P and $a \notin U$, or a, b, c belong to three different parts in P.
  - $w_{f3}(P)$ = the number of input forbidden fan triplets $\neg(a|b|c)$ such that two elements in $\{a, b, c\}$ belong to two different parts in P and the remaining one does not belong to U.
- Let $I(T)$ be the set of internal nodes in a tree T. For $v \in I(T)$, denote: T_v = the subtree of T rooted at v, $\{v_1, \ldots, v_l\}$ = the set of children of v, and π_v = the partition of $\Lambda(T_v)$ into $\Lambda(T_{v_1}), \ldots, \Lambda(T_{v_l})$.

Lemma: For any T with $\Lambda(T) = S$, the # of input constraints satisfied by T is: $\sum_{v \in I(T)}(w_2(\pi_v) + w_3(\pi_v) + w_{f2}(\pi_v) + w_{f3}(\pi_v))$.

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

- Proceed as in Wu's algorithm, but for each $U \subseteq S$, define $score(U) = \max_{\ell=2}^{k}\{score_\ell(U)\}$, where $score_\ell(U) =$

$$\max_{\ell-\text{partition } U_1,\ldots,U_\ell \text{ of } U}\Big\{\sum_{i=1}^{\ell} score(U_i) + \sum_{j=2}^{3}\big(w_j(U_1,\ldots,U_\ell) + w_{fj}(U_1,\ldots,U_\ell)\big)\Big\}.$$

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

- Proceed as in Wu's algorithm, but for each $U \subseteq S$, define $score(U) = \max_{\ell=2}^{k}\{score_\ell(U)\}$, where $score_\ell(U) =$

$$\max_{\ell-\text{partition } U_1,\ldots,U_\ell \text{ of } U}\Big\{\sum_{i=1}^{\ell} score(U_i) + \sum_{j=2}^{3}\big(w_j(U_1,\ldots,U_\ell) + w_{fj}(U_1,\ldots,U_\ell)\big)\Big\}.$$

- By the Lemma, $score(U)$ = the maximum number of input constraints that can be consistent with a k-ary tree leaf-labeled by U.

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

- Proceed as in Wu's algorithm, but for each $U \subseteq S$, define $score(U) = \max_{\ell=2}^{k}\{score_\ell(U)\}$, where $score_\ell(U) =$

$$\max_{\ell-\text{partition } U_1,\ldots,U_\ell \text{ of } U}\Big\{\sum_{i=1}^{\ell} score(U_i) + \sum_{j=2}^{3}\big(w_j(U_1,\ldots,U_\ell) + w_{fj}(U_1,\ldots,U_\ell)\big)\Big\}.$$

- By the Lemma, $score(U)$ = the maximum number of input constraints that can be consistent with a k-ary tree leaf-labeled by U.

- Total # of partitions considered $\leq \sum_{q=1}^{n}\binom{n}{q}\sum_{\ell=2}^{k}\ell^q \leq \sum_{\ell=2}^{k}(\ell+1)^n \leq (k+1)^{n+1}$.

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

- Proceed as in Wu's algorithm, but for each $U \subseteq S$, define $score(U) = \max_{\ell=2}^{k}\{score_\ell(U)\}$, where $score_\ell(U) =$

$$\max_{\ell-\text{partition } U_1,\ldots,U_\ell \text{ of } U}\Big\{\sum_{i=1}^{\ell} score(U_i) + \sum_{j=2}^{3}\big(w_j(U_1,\ldots,U_\ell) + w_{fj}(U_1,\ldots,U_\ell)\big)\Big\}.$$

- By the Lemma, $score(U)$ = the maximum number of input constraints that can be consistent with a k-ary tree leaf-labeled by U.

- Total # of partitions considered $\leq \sum_{q=1}^{n}\binom{n}{q}\sum_{\ell=2}^{k}\ell^q \leq \sum_{\ell=2}^{k}(\ell+1)^n \leq (k+1)^{n+1}$.

  $O(m+n)$ time to compute $w_2(P), w_3(P), w_{f2}(P), w_{f3}(P)$ for each partition P.

2. An exact algorithm for MTC, cont.

- Let $k \geq 2$ be a parameter.

  The new algorithm forces the output to be a k-ary tree, i.e., a tree in which every internal node has at most k children.

- Proceed as in Wu's algorithm, but for each $U \subseteq S$, define $score(U) = \max_{\ell=2}^{k}\{score_\ell(U)\}$, where $score_\ell(U) =$

$$\max_{\ell-\text{partition } U_1,\ldots,U_\ell \text{ of } U}\Big\{\sum_{i=1}^{\ell} score(U_i) + \sum_{j=2}^{3}\big(w_j(U_1,\ldots,U_\ell) + w_{fj}(U_1,\ldots,U_\ell)\big)\Big\}.$$

- By the Lemma, $score(U)$ = the maximum number of input constraints that can be consistent with a k-ary tree leaf-labeled by U.

- Total # of partitions considered $\leq \sum_{q=1}^{n}\binom{n}{q}\sum_{\ell=2}^{k}\ell^q \leq \sum_{\ell=2}^{k}(\ell+1)^n \leq (k+1)^{n+1}$.

  $O(m+n)$ time to compute $w_2(P), w_3(P), w_{f2}(P), w_{f3}(P)$ for each partition P.

  $\Rightarrow$ $\boxed{O((k+1)^{n+1}(m+n))}$ time to solve MTC exactly over all k-ary trees.

3. An ETAS for MTC

Exponential-time approximation scheme (ETAS) for MTC:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in $O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m + n))$ time.

3. An ETAS for MTC

Exponential-time approximation scheme (ETAS) for MTC:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in

$O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m + n))$ time.

Obtained by selecting $k := \lceil \frac{12}{\epsilon} \rceil$ and applying the exact algorithm for optimal k-ary trees above. The correctness follows from:

3. An ETAS for MTC

Exponential-time approximation scheme (ETAS) for MTC:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in

$O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m + n))$ time.

Obtained by selecting $k := \lceil \frac{12}{\epsilon} \rceil$ and applying the exact algorithm for optimal k-ary trees above. The correctness follows from:

Theorem: For any tree T, there exists a k-ary tree T' with $\Lambda(T') = \Lambda(T)$ that is consistent with at least a fraction of $(1 - 12/k)$ of the input constraints consistent with T.

3. An ETAS for MTC

Exponential-time approximation scheme (ETAS) for MTC:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in

$O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m + n))$ time.

Obtained by selecting $k := \lceil \frac{12}{\epsilon} \rceil$ and applying the exact algorithm for optimal k-ary trees above. The correctness follows from:

Theorem: For any tree T, there exists a k-ary tree T' with $\Lambda(T') = \Lambda(T)$ that is consistent with at least a fraction of $(1 - 12/k)$ of the input constraints consistent with T.

Proof sketch: Use a probabilistic argument to show that there exists such a k-ary tree obtained from T by replacing each node with degree $> k$ by a k-ary subtree.

3. An ETAS for MTC

Exponential-time approximation scheme (ETAS) for MTC:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in

$O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m+n))$ time.

Obtained by selecting $k := \lceil \frac{12}{\epsilon} \rceil$ and applying the exact algorithm for optimal k-ary trees above. The correctness follows from:

Theorem: For any tree T, there exists a k-ary tree T' with $\Lambda(T') = \Lambda(T)$ that is consistent with at least a fraction of $(1 - 12/k)$ of the input constraints consistent with T.

Proof sketch: Use a probabilistic argument to show that there exists such a k-ary tree obtained from T by replacing each node with degree $> k$ by a k-ary subtree.

- Suppose $\deg(v) = \ell > k$. For any fan triplet $a|b|c$ contributing to $w_3(\pi_v)$, if a, b, c are distributed among k groups uniformly at random then $Pr[$any two elements in $\{a, b, c\}$ end up in the same group$] \leq 1/k + 2/k = 3/k$.

  $\Rightarrow$ There exists some partition of $v_1, \ldots, v_\ell$ into k subtrees that satisfies $\geq 1 - 3/k$ of the fan triplets contributing to $w_3(\pi_v)$.

  etc. □

4. A PTAS for dense MTC

Byrka, Gawrychowski, Huber, Kelk (2010): MTC is APX-hard in the general case.
$\Rightarrow$ Unlikely that a polynomial-time approximation scheme (PTAS) exists.

4. A PTAS for dense MTC

Byrka, Gawrychowski, Huber, Kelk (2010): MTC is APX-hard in the general case.
$\Rightarrow$ Unlikely that a polynomial-time approximation scheme (PTAS) exists.

However, we can get a PTAS for dense MTC, where an instance is called dense if it contains $\Omega(n^3)$ constraints:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in polyn. time.

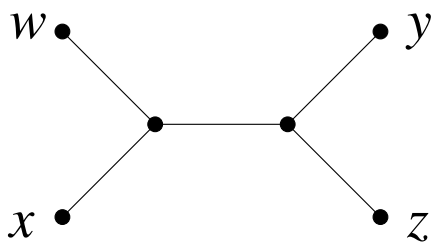
4. A PTAS for dense MTC

Byrka, Gawrychowski, Huber, Kelk (2010): MTC is APX-hard in the general case.
$\Rightarrow$ Unlikely that a polynomial-time approximation scheme (PTAS) exists.

However, we can get a PTAS for dense MTC, where an instance is called dense if it contains $\Omega(n^3)$ constraints:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in polyn. time.

Relies on the technique introduced by [Jiang, Kearney, Li (2001)] for maximizing *unrooted quartet consistency* (MQC).

Unrooted quartet:



Jiang *et al.* (2001) developed a PTAS for dense MQC, where the input consists of $\Omega(n^4)$ unrooted quartets.

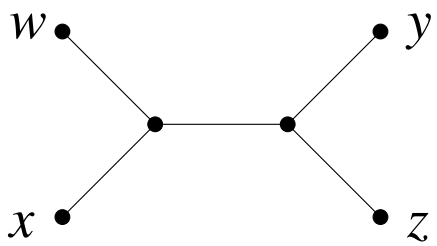
4. A PTAS for dense MTC

Byrka, Gawrychowski, Huber, Kelk (2010): MTC is APX-hard in the general case.
$\Rightarrow$ Unlikely that a polynomial-time approximation scheme (PTAS) exists.

However, we can get a PTAS for dense MTC, where an instance is called dense if it contains $\Omega(n^3)$ constraints:
For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1 - \epsilon)$ of the maximum number of input constraints consistent with any tree in polyn. time.

Relies on the technique introduced by [Jiang, Kearney, Li (2001)] for maximizing *unrooted quartet consistency* (MQC).

Unrooted quartet:



Jiang *et al.* (2001) developed a PTAS for dense MQC, where the input consists of $\Omega(n^4)$ unrooted quartets.

We cannot apply their PTAS directly to our problem because of the fan triplets and the forbidden rooted triplets, but we can adapt their technique to our setting.

4. A PTAS for dense MTC, cont.

The resulting algorithm:

- Given $\epsilon > 0$, let k be some constant that depends on $1/\epsilon$.
  (A suitable value for k will be derived later.)

4. A PTAS for dense MTC, cont.

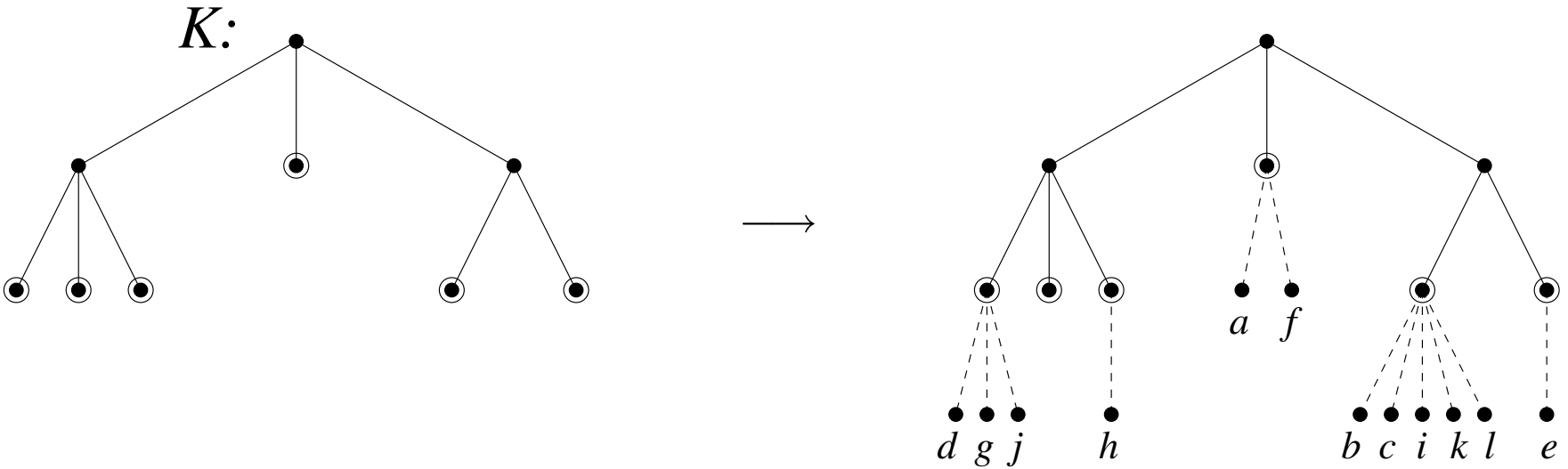
The resulting algorithm:

- Given $\epsilon > 0$, let k be some constant that depends on $1/\epsilon$.
  (A suitable value for k will be derived later.)
- Generate all unlabeled, unordered rooted trees with k leaves and no degree-1 nodes.

4. A PTAS for dense MTC, cont.

The resulting algorithm:

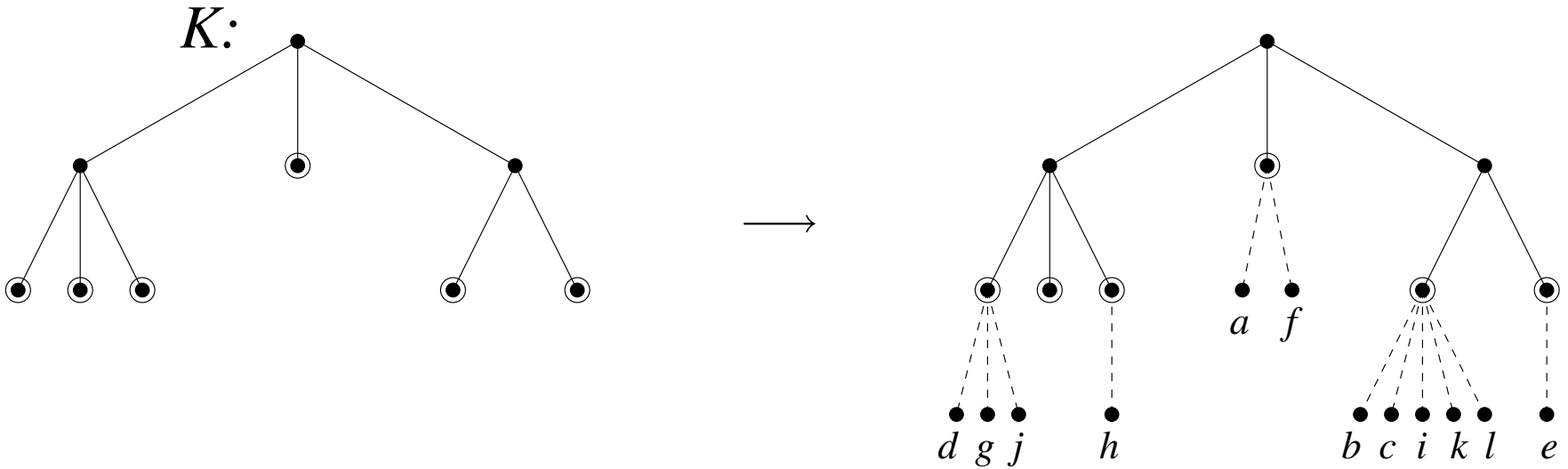
- Given $\epsilon > 0$, let k be some constant that depends on $1/\epsilon$.
  (A suitable value for k will be derived later.)
- Generate all unlabeled, unordered rooted trees with k leaves and no degree-1 nodes.
- For each such tree K: Every leaf of K is called a bin.
  Approximately solve the *Label-to-Bin Assignment Problem* (LBA) on K:
  Attach the n leaf labels to the k bins of K so that each bin gets $\leq \frac{6n}{k}$ leaves and the resulting tree is consistent with the maximum # of input constraints.



4. A PTAS for dense MTC, cont.

The resulting algorithm:

- Given $\epsilon > 0$, let k be some constant that depends on $1/\epsilon$.
  (A suitable value for k will be derived later.)
- Generate all unlabeled, unordered rooted trees with k leaves and no degree-1 nodes.
- For each such tree K: Every leaf of K is called a bin.
  Approximately solve the *Label-to-Bin Assignment Problem* (LBA) on K:
  Attach the n leaf labels to the k bins of K so that each bin gets $\leq \frac{6n}{k}$ leaves and the resulting tree is consistent with the maximum # of input constraints.



- Output the best solution to MTC found.

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

- Let I be the set of input triplet constraints and let $\mathcal{R}_K$ be the set of all rooted triplets over the bins consistent with K.

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

- Let I be the set of input triplet constraints and let $\mathcal{R}_K$ be the set of all rooted triplets over the bins consistent with K.
- Variable $x_{\ell i} = 1$ if leaf label ℓ is assigned to bin i, and $= 0$ otherwise.

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

- Let I be the set of input triplet constraints and let $\mathcal{R}_K$ be the set of all rooted triplets over the bins consistent with K.
- Variable $x_{\ell i} = 1$ if leaf label ℓ is assigned to bin i, and $= 0$ otherwise.
- For every resolved triplet $ab|c \in I$, define the polynomial:

$$p_{ab|c}(x) = \sum_{ij|k \in \mathcal{R}_K} x_{ai}x_{bj}x_{ck} + x_{bi}x_{aj}x_{ck}$$

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

- Let I be the set of input triplet constraints and let $\mathcal{R}_K$ be the set of all rooted triplets over the bins consistent with K.
- Variable $x_{\ell i} = 1$ if leaf label ℓ is assigned to bin i, and $= 0$ otherwise.
- For every resolved triplet $ab|c \in I$, define the polynomial:

$$p_{ab|c}(x) = \sum_{ij|k \in \mathcal{R}_K} x_{ai}x_{bj}x_{ck} + x_{bi}x_{aj}x_{ck}$$

- For every fan triplet $a|b|c \in I$, define:

$$p_{a|b|c}(x) = \sum_{i|j|k \in \mathcal{R}_K} x_{ai}x_{bj}x_{ck} + x_{ai}x_{cj}x_{bk} + \ldots + x_{ci}x_{bj}x_{ak}$$

4. A PTAS for dense MTC, cont.

To approximately solve the Label-to-Bin Assignment Problem (LBA) on K, we follow the approach of Jiang *et al.* (2001) and formulate it as an integer program:

- Let I be the set of input triplet constraints and let $\mathcal{R}_K$ be the set of all rooted triplets over the bins consistent with K.
- Variable $x_{\ell i} = 1$ if leaf label ℓ is assigned to bin i, and $= 0$ otherwise.
- For every resolved triplet $ab|c \in I$, define the polynomial:

$$p_{ab|c}(x) = \sum_{ij|k \in \mathcal{R}_K} x_{ai}x_{bj}x_{ck} + x_{bi}x_{aj}x_{ck}$$

- For every fan triplet $a|b|c \in I$, define:

$$p_{a|b|c}(x) = \sum_{i|j|k \in \mathcal{R}_K} x_{ai}x_{bj}x_{ck} + x_{ai}x_{cj}x_{bk} + \ldots + x_{ci}x_{bj}x_{ak}$$

- For every forbidden resolved triplet $\neg(ab|c) \in I$, define:

$$p_{\neg(ab|c)}(x) = p_{ac|b}(x) + p_{bc|a}(x) + p_{a|b|c}(x)$$

- For every forbidden fan triplet $\neg(a|b|c) \in I$, define:

$$p_{\neg(a|b|c)}(x) = p_{ab|c}(x) + p_{ac|b}(x) + p_{bc|a}(x)$$

4. A PTAS for dense MTC, cont.

- Finally, define:

$$p(x) = \sum_{ab|c \in I} p_{ab|c}(x) + \sum_{a|b|c \in I} p_{a|b|c}(x) + \sum_{\neg(ab|c) \in I} p_{\neg(ab|c)}(x) + \sum_{\neg(a|b|c) \in I} p_{\neg(a|b|c)}(x)$$

- LBA becomes:

Maximize $p(x)$ subject to

$\sum_{i=1}^{k} x_{\ell i} = 1$ for each leaf label ℓ, $\sum_{\ell=1}^{n} x_{\ell i} \leq \frac{6n}{k}$ for each bin i,

and $x_{\ell i} \in \{0, 1\}$ for all ℓ, i.

4. A PTAS for dense MTC, cont.

- Finally, define:

$$p(x) = \sum_{ab|c \in I} p_{ab|c}(x) + \sum_{a|b|c \in I} p_{a|b|c}(x) + \sum_{\neg(ab|c) \in I} p_{\neg(ab|c)}(x) + \sum_{\neg(a|b|c) \in I} p_{\neg(a|b|c)}(x)$$

- LBA becomes:

Maximize $p(x)$ subject to

$\sum_{i=1}^{k} x_{\ell i} = 1$ for each leaf label ℓ, $\sum_{\ell=1}^{n} x_{\ell i} \leq \frac{6n}{k}$ for each bin i,

and $x_{\ell i} \in \{0, 1\}$ for all ℓ, i.

Arora, Karger, Karpinski (1999):
A degree-d polynomial integer program is c-smooth if the absolute value of each coefficient of each degree-i term is $\leq c \cdot n^{d-i}$.

4. A PTAS for dense MTC, cont.

- Finally, define:

$$p(x) = \sum_{ab|c \in I} p_{ab|c}(x) + \sum_{a|b|c \in I} p_{a|b|c}(x) + \sum_{\neg(ab|c) \in I} p_{\neg(ab|c)}(x) + \sum_{\neg(a|b|c) \in I} p_{\neg(a|b|c)}(x)$$

- LBA becomes:

Maximize $p(x)$ subject to
$\sum_{i=1}^{k} x_{\ell i} = 1$ for each leaf label ℓ, $\sum_{\ell=1}^{n} x_{\ell i} \leq \frac{6n}{k}$ for each bin i,
and $x_{\ell i} \in \{0, 1\}$ for all ℓ, i.

Arora, Karger, Karpinski (1999):
A degree-d polynomial integer program is c-smooth if the absolute value of each coefficient of each degree-i term is $\leq c \cdot n^{d-i}$.

$\Rightarrow$ The above is a c-smooth degree-3 polynomial integer program, where $c = O(1)$.

4. A PTAS for dense MTC, cont.

- Finally, define:

$$p(x) = \sum_{ab|c \in I} p_{ab|c}(x) + \sum_{a|b|c \in I} p_{a|b|c}(x) + \sum_{\neg(ab|c) \in I} p_{\neg(ab|c)}(x) + \sum_{\neg(a|b|c) \in I} p_{\neg(a|b|c)}(x)$$

- LBA becomes:

Maximize $p(x)$ subject to
$\sum_{i=1}^{k} x_{\ell i} = 1$ for each leaf label ℓ, $\sum_{\ell=1}^{n} x_{\ell i} \leq \frac{6n}{k}$ for each bin i,
and $x_{\ell i} \in \{0, 1\}$ for all ℓ, i.

Arora, Karger, Karpinski (1999):
A degree-d polynomial integer program is c-smooth if the absolute value of each coefficient of each degree-i term is $\leq c \cdot n^{d-i}$.

$\Rightarrow$ The above is a c-smooth degree-3 polynomial integer program, where $c = O(1)$.

As in Jiang *et al.* (2001), apply the general-purpose PTAS of Arora *et al.* (1999) for c-smooth integer programs to get a PTAS for our variant of LBA.

4. A PTAS for dense MTC, cont.

Time complexity analysis:

- k is a constant, so the number of unlabeled, unordered rooted trees with k leaves and no degree-1 nodes is a constant.

4. A PTAS for dense MTC, cont.

Time complexity analysis:

- k is a constant, so the number of unlabeled, unordered rooted trees with k leaves and no degree-1 nodes is a constant.
- For each such tree, the algorithm constructs an LBA instance and applies the PTAS of Arora *et al.* (1999) to approximately solve it.

4. A PTAS for dense MTC, cont.

Time complexity analysis:

- k is a constant, so the number of unlabeled, unordered rooted trees with k leaves and no degree-1 nodes is a constant.
- For each such tree, the algorithm constructs an LBA instance and applies the PTAS of Arora *et al.* (1999) to approximately solve it.

$\Rightarrow$ The time complexity is polynomial. □

4. A PTAS for dense MTC, cont.

Approximation ratio analysis:

Lemma: Let T be a tree consistent with the maximum number of constraints in I. There exists a tree T' obtained by attaching the n leaf labels to an unlabeled tree with k bins such that $|\mathcal{R}_{T'} \cap I| \geq |\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3$.

4. A PTAS for dense MTC, cont.

Approximation ratio analysis:

Lemma: Let T be a tree consistent with the maximum number of constraints in I. There exists a tree T' obtained by attaching the n leaf labels to an unlabeled tree with k bins such that $|\mathcal{R}_{T'} \cap I| \geq |\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3$.

The PTAS of Arora *et al.* finds an $(1 - \epsilon'')$ approximation T'' of such a T', so:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1 - \epsilon'') \cdot |\mathcal{R}_{T'} \cap I| \;\geq\; (1 - \epsilon'') \cdot (|\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3)$$

4. A PTAS for dense MTC, cont.

Approximation ratio analysis:

Lemma: Let T be a tree consistent with the maximum number of constraints in I. There exists a tree T' obtained by attaching the n leaf labels to an unlabeled tree with k bins such that $|\mathcal{R}_{T'} \cap I| \geq |\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3$.

The PTAS of Arora *et al.* finds an $(1 - \epsilon'')$ approximation T'' of such a T', so:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1-\epsilon'') \cdot |\mathcal{R}_{T'} \cap I| \;\geq\; (1-\epsilon'') \cdot (|\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3)$$

Next, note that $|I| \geq \gamma \cdot n^3$ for some constant γ by the definition of dense.
The $\frac{1}{4}$-approximation algorithm from before $\Rightarrow |\mathcal{R}_T \cap I| \geq \frac{|I|}{4} \Rightarrow n^3 \leq \frac{4 \cdot |\mathcal{R}_T \cap I|}{\gamma}$

4. A PTAS for dense MTC, cont.

Approximation ratio analysis:

Lemma: Let T be a tree consistent with the maximum number of constraints in I. There exists a tree T' obtained by attaching the n leaf labels to an unlabeled tree with k bins such that $|\mathcal{R}_{T'} \cap I| \geq |\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3$.

The PTAS of Arora *et al.* finds an $(1 - \epsilon'')$ approximation T'' of such a T', so:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1 - \epsilon'') \cdot |\mathcal{R}_{T'} \cap I| \;\geq\; (1 - \epsilon'') \cdot (|\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3)$$

Next, note that $|I| \geq \gamma \cdot n^3$ for some constant γ by the definition of dense.
The $\frac{1}{4}$-approximation algorithm from before $\Rightarrow |\mathcal{R}_T \cap I| \geq \frac{|I|}{4} \Rightarrow n^3 \leq \frac{4 \cdot |\mathcal{R}_T \cap I|}{\gamma}$

Combine the inequalities:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1 - \epsilon'') \cdot |\mathcal{R}_T \cap I| \cdot (1 - \frac{96}{k} \cdot \frac{4}{\gamma})$$

4. A PTAS for dense MTC, cont.

Approximation ratio analysis:

Lemma: Let T be a tree consistent with the maximum number of constraints in I. There exists a tree T' obtained by attaching the n leaf labels to an unlabeled tree with k bins such that $|\mathcal{R}_{T'} \cap I| \geq |\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3$.

The PTAS of Arora *et al.* finds an $(1-\epsilon'')$ approximation T'' of such a T', so:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1-\epsilon'') \cdot |\mathcal{R}_{T'} \cap I| \;\geq\; (1-\epsilon'') \cdot (|\mathcal{R}_T \cap I| - \frac{96}{k} \cdot n^3)$$

Next, note that $|I| \geq \gamma \cdot n^3$ for some constant γ by the definition of dense.
The $\frac{1}{4}$-approximation algorithm from before $\Rightarrow |\mathcal{R}_T \cap I| \geq \frac{|I|}{4} \Rightarrow n^3 \leq \frac{4 \cdot |\mathcal{R}_T \cap I|}{\gamma}$

Combine the inequalities:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; (1-\epsilon'') \cdot |\mathcal{R}_T \cap I| \cdot (1 - \frac{96}{k} \cdot \frac{4}{\gamma})$$

Finally, for any specified $\epsilon > 0$, choosing $k \geq \frac{96 \cdot 4 \cdot 2}{\gamma \cdot \epsilon}$ and $\epsilon'' \leq \frac{\epsilon}{2}$ gives:

$$|\mathcal{R}_{T''} \cap I| \;\geq\; |\mathcal{R}_T \cap I| \cdot (1 - \frac{\epsilon}{2}) \cdot (1 - \frac{\epsilon}{2}) \;\geq\; |\mathcal{R}_T \cap I| \cdot (1-\epsilon) \qquad \square$$

Summary

1. Polynomial-time $\frac{1}{4}$-approximation algorithm for MTC.

2. Extension of Wu's exact exponential-time algorithm for MTC restricted to resolved triplets to also allow fan triplets, forbidden resolved triplets, and forbidden fan triplets.

   It takes an additional parameter $k \geq 2$ as input and forces the output to be a tree in which every internal node has at most k children.

   $\Rightarrow$ Running time: $\boxed{O((k+1)^{n+1}(m+n))}$

3. Exponential-time approximation scheme (ETAS) for MTC:

   For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in $\boxed{O((\lceil \frac{12}{\epsilon} \rceil + 1)^{n+1}(m+n))}$ time.

4. Polynomial-time approximation scheme (PTAS) for dense MTC, where an instance is called dense if it contains $\Omega(n^3)$ constraints:

   For any constant $\epsilon > 0$, we can build a tree consistent with $\geq (1-\epsilon)$ of the maximum number of input constraints consistent with any tree in polyn. time.