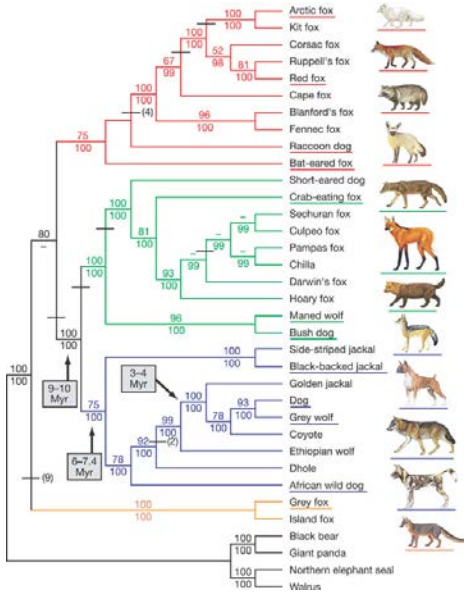


Nature 438, pp. 803-819 (2005)

Label Isomorphism: On the MAF Problems for Multiple Trees

Jianer Chen

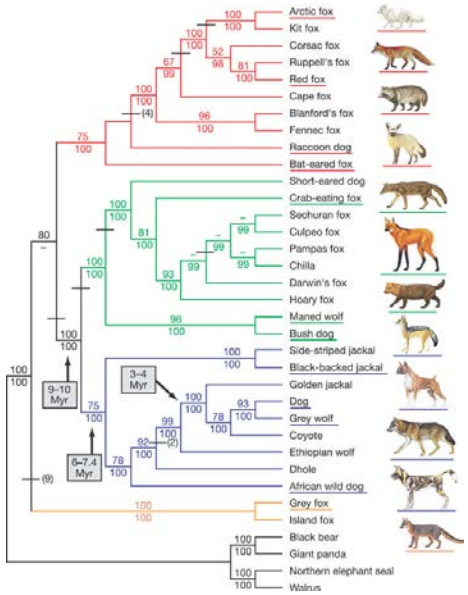
School of Computer Science and Educational Software

Guangzhou University, P.R. China

Department of Computer Science & Engineering

Texas A&M University, USA

IMS Singapore 2017



Nature 438, pp. 803-819 (2005)

A Quick Overview on MAF Algorithms and Label Isomorphism: On the MAF Problems for Multiple Trees

Jianer Chen

School of Computer Science and Educational Software

Guangzhou University, P.R. China

Department of Computer Science & Engineering

Texas A&M University, USA

IMS Singapore 2017

Definitions

- **X-Tree (X-forest)**

  A tree (forest) T for which there is a one-to-one mapping from the leaves of T to a symbol set X.

- **Agreement forest**

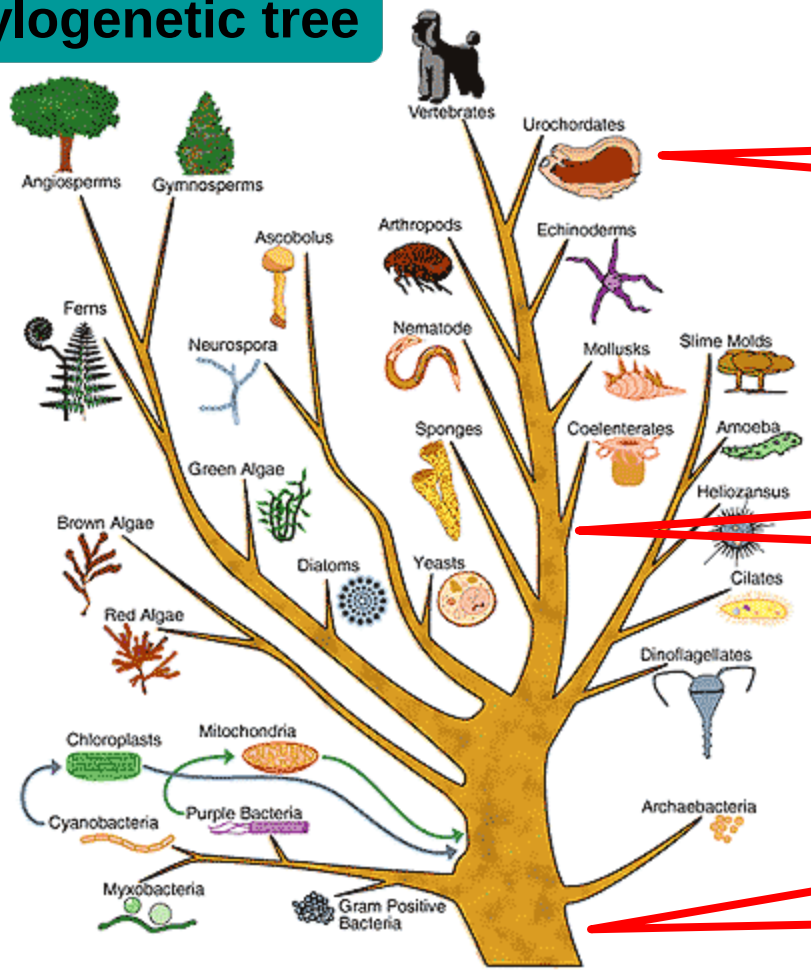
  An agreement forest of two X-trees is an X-forest that is a subgraph of both the X-trees.

- **The Maximum Agreement Forest (MAF) Problem**

  Find a maximum agreement forest (maf) for two given X-trees

Background

Phylogenetic tree



www.bio.Miami.edu

leaves are labeled with extant species

internal nodes correspond to speciation events

if rooted, the root represents the ancestor of all the species

Background

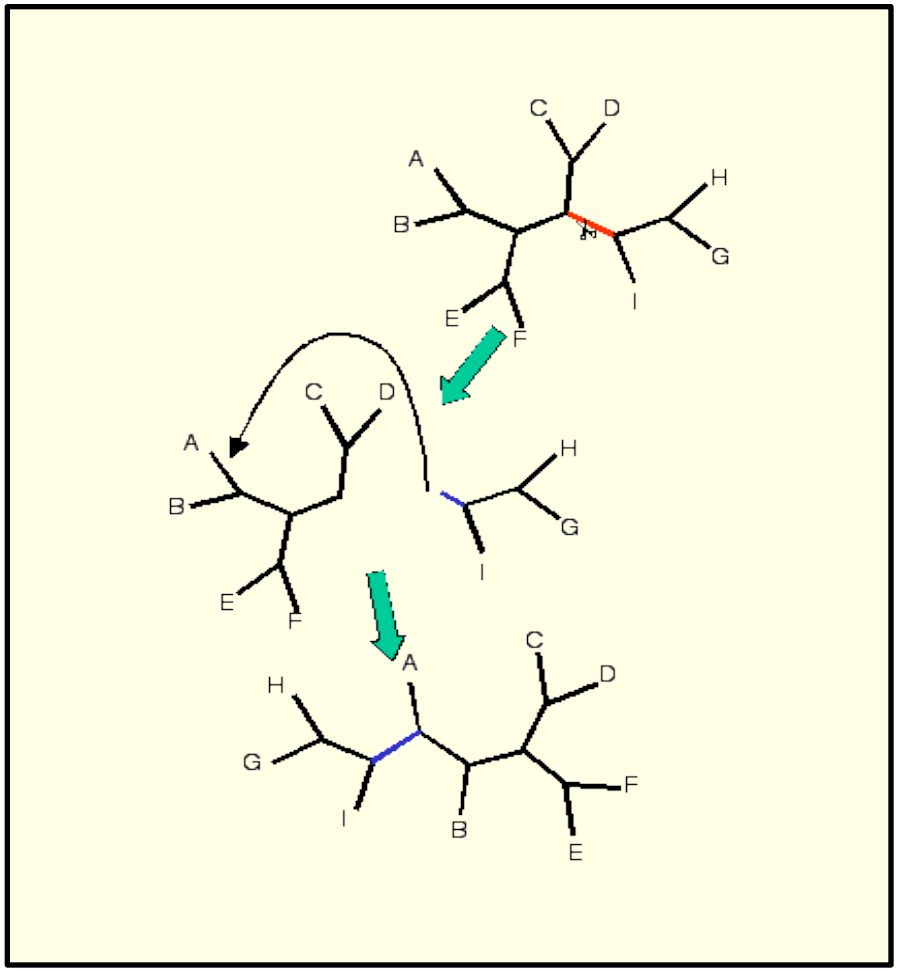
- Different experiments may result in different phylogenetic trees
  - Morphology
  - Molecular biology

Background

- **Different experiments may result in different phylogenetic trees**
  - **Morphology**
  - **Molecular biology**
- **Distance metrics have been proposed to facilitate the comparison of different phylogenetic trees for their similarity:**

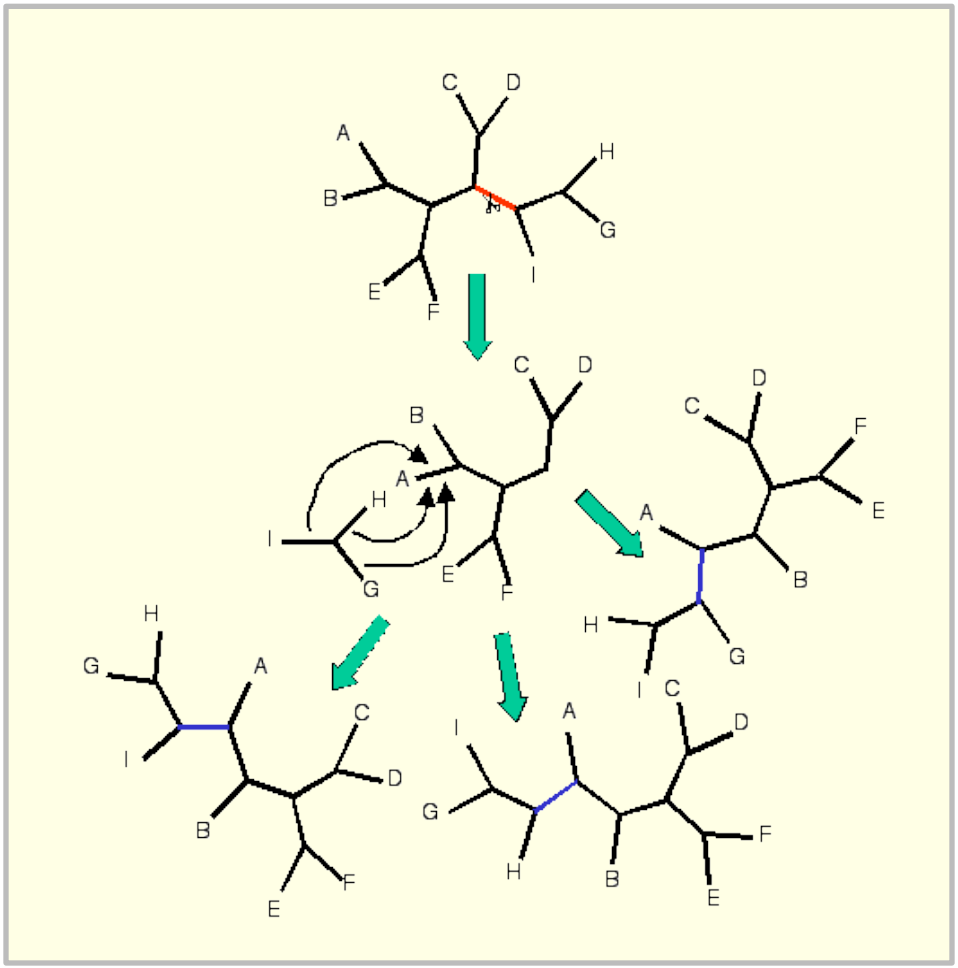
  Robinson-Foulds, Nearest Neighbor Interchange (NNI), TBR, SPR, …

Subtree-Prune-and-Regraft (SPR)



http://artedi.ebc.uu.se/course/X3-2004/Phylogeny/

Tree-Bisection-and-Reconnection (TBR)



http://artedi.ebc.uu.se/course/X3-2004/Phylogeny/

Background

- **The Maximum Agreement Forest (MAF) Problem**
  Find a maximum agreement forest (maf) for two given X-trees
  - **SPR distance corresponds to the order of an MAF in rooted trees [Bordewich 2005]**
  - **TBR distance corresponds to the order of an MAF in unrooted trees [Allen 2000]**

Background

- **The Maximum Agreement Forest (MAF) Problem**
  Find a maximum agreement forest (maf) for two given X-trees
  - **SPR distance corresponds to the order of an MAF in rooted trees [Bordewich 2005]**
  - **TBR distance corresponds to the order of an MAF in unrooted trees [Allen 2000]**
- **The MAF problem can be equivalently defined as to remove the minimum number of edges to make the two X-trees isomorphic (thus, a minimization problem, instead of a maximization problem).**

Background

- **The Maximum Agreement Forest (MAF) Problem**
  Find a maximum agreement forest (maf) for two given X-trees
  - **SPR distance corresponds to the order of an MAF in rooted trees [Bordewich 2005]**
  - **TBR distance corresponds to the order of an MAF in unrooted trees [Allen 2000]**
- **The MAF problem can be equivalently defined as to remove the minimum number of edges to make the two X-trees isomorphic (thus, a minimization problem, instead of a maximization problem).**
- **The problem of constructing an MAF for two binary X-trees is NP-hard [Allen and Steel 2001], and MAX SNP-hard [Bordewich et al. 2008]**

Background

- Most existing algorithms work by progressively removing edges from the X-trees to eventually make the trees isomorphic. Thus, we are actually working on two X-forests, instead of two X-trees.

Background

- Most existing algorithms work by progressively removing edges from the X-trees to eventually make the trees isomorphic. Thus, we are actually working on two X-forests, instead of two X-trees.

- **The MAF Problem Revised**
  Remove the minimum number of edges from two given X-forests to make them isomorphic.

Background

- Most existing algorithms work by progressively removing edges from the X-trees to eventually make the trees isomorphic. Thus, we are actually working on two X-forests, instead of two X-trees.

- **The MAF Problem Revised**
  Remove the minimum number of edges from two given X-forests to make them isomorphic.

- **The MAF Problem Further Revised (parameterized)**
  For two given X-forests, can we remove k edges from one of the X-forests to make it a subgraph of the other X-forest.

Background

- **Isomorphism of X-forests**

  The definition is up to a "forced contraction" operation that removes unlabeled leaves and contracts degree-2 vertices.

- **Forest-structured set**

  An X-forest can be regarded as the set X plus a structure enforced by the forest. The structure restricts the ways of splitting the set X. This view is more formal and sometimes more convenient when discussing operations on X-forests.

Parameterized Algorithms

For two unrooted binary trees:

- **Fixed parameter tractable [Allen-Steel 2001]**
- **Time $O(4^k k^5 + n^{O(1)})$ [Hallett-McCartin 2007]**
- **$O^*(4^k k)$-time algorithm [Whidden-Zeh 2009]**
- **$O^*(3^k)$-time algorithm [JC-Fan-Sze 2015]**

Parameterized Algorithms

For two unrooted binary trees:

- **Fixed parameter tractable [Allen-Steel 2001]**
- **Time $O(4^k k^5 + n^{O(1)})$ [Hallett-McCartin 2007]**
- **$O^*(4^k k)$-time algorithm [Whidden-Zeh 2009]**
- **$O^*(3^k)$-time algorithm [JC-Fan-Sze 2015]**

For two rooted binary trees:

- **Time $O(4^k k^4 + n^3)$ [Bordewich et al. 2008]**
- **$O^*(2.42^k)$-time algorithm [Whidden et al. 2013]**
- **$O^*(2.344^k)$-time algorithm [Z. Chen-Fan-Wang 2013]**

Call for work

- **"We should also look at the case where T_1 and T_2 are not necessarily binary unrooted trees."**

  [Hallett-McCartin, *Theory of Computing Systems* 41, pp. 539-550, 2007]

- **"The most important open problem is extending our approach to computing MAFs and MAAFs for multifurcating trees and for more than two trees."**

  [Whidden-Beiko-Zeh, *SIAM J. Comput*. 42(4), pp. 1431-1466, 2013]

Call for work

- **"We should also look at the case where T_1 and T_2 are not necessarily binary unrooted trees."**

  [Hallett-McCartin, *Theory of Computing Systems* 41, pp. 539-550, 2007]

- **"The most important open problem is extending our approach to computing MAFs and MAAFs for multifurcating trees and for more than two trees."**

  [Whidden-Beiko-Zeh, *SIAM J. Comput*. 42(4), pp. 1431-1466, 2013]

Parameterized algorithms for

- **two general X-trees**
- **Multiple binary X-trees**
- **Multiple general X-trees**

Call for work

Remark. It makes perfect sense to consider MAF for non-binary trees and for more than two trees:

- **Ambiguities in which the order of more than 2 branches cannot be reliably resolved by phylogenetic tree construction algorithms (soft multifurcations)**
- **Truly multifurcations (hard multifurcations)**
- **NCBI Taxonomy (an important source of species trees): more than half of its branches being multifurcating**
- **MAF on multifurcating trees also corresponds to SPR and TBR distances on the trees**
- **Phylogenetic tree for the same collection of species may be constructed using more than two methods, each producing a different tree.**

Two General X-trees (quick overview)

Given a pair of general X-forests (F_1, F_2), and an integer k, remove at most k edges in the larger-order X-forest so that it becomes a subgraph of F_1 and F_2.

Two General X-trees (quick overview)

Given a pair of general X-forests (F_1, F_2), and an integer k, remove at most k edges in the larger-order X-forest so that it becomes a subgraph of F_1 and F_2.

General Idea (for most proposed algorithms):

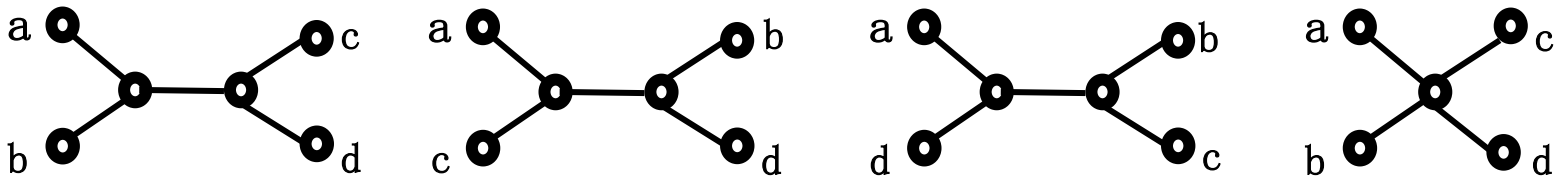
Repeat:

1. Identify structure inconsistency in F_1 and F_2;
2. Pick a collection B of edges in F_1 (and in F_2), and make sure one of the edges in B must be deleted in order to remove the inconsistency;
3. Branch on removing each of the edges in the collection B.

Two General X-trees (quick overview)

Branching based on global inconsistency

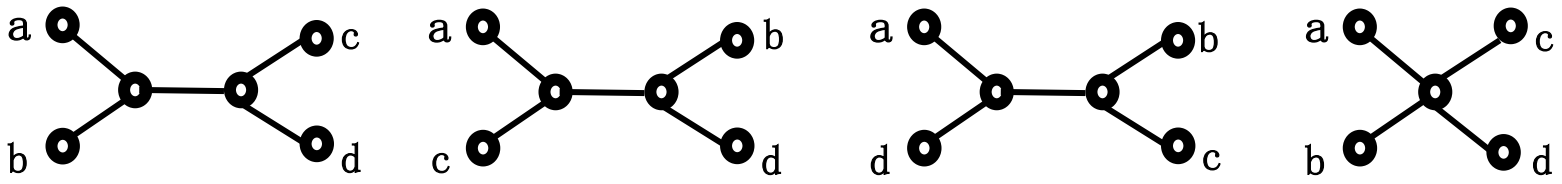
quartet on four labels a, b, c, d



Two General X-trees (quick overview)

Branching based on global inconsistency

quartet on four labels a, b, c, d



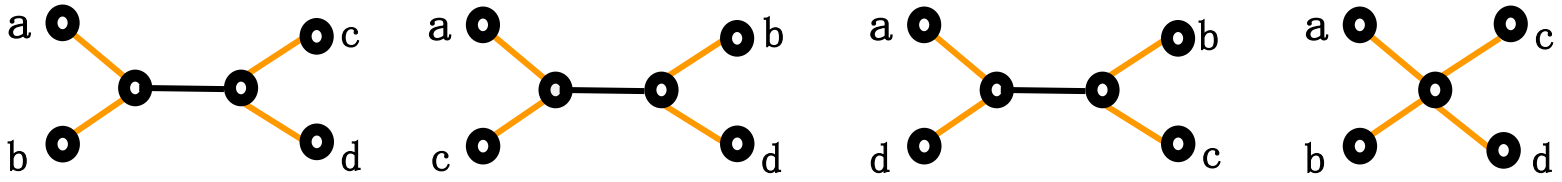
Observation.

If there are four labels a, b, c, d for which the quartets in F_1 and F_2 are different, then one of the edges in the quartet in F_1 is not in any MAF of F_1 and F_2.

Two General X-trees (quick overview)

Branching based on global inconsistency

quartet on four labels a, b, c, d



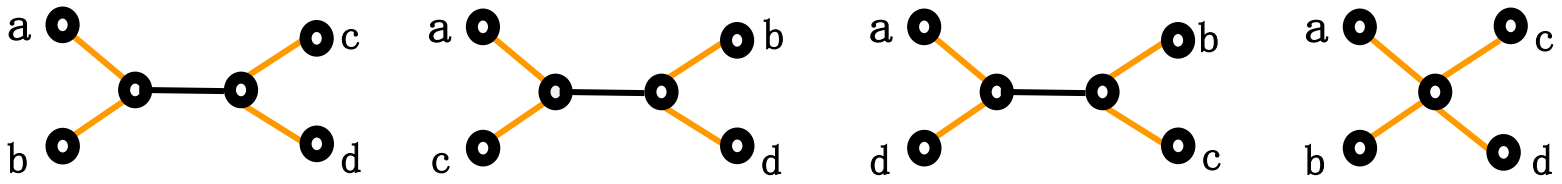
Observation.

If there are four labels a, b, c, d for which the quartets in F_1 and F_2 are different, then one of the edges in the quartet in F_1 is not in any MAF of F_1 and F_2.

Two General X-trees (quick overview)

Branching based on global inconsistency

quartet on four labels a, b, c, d



Observation.

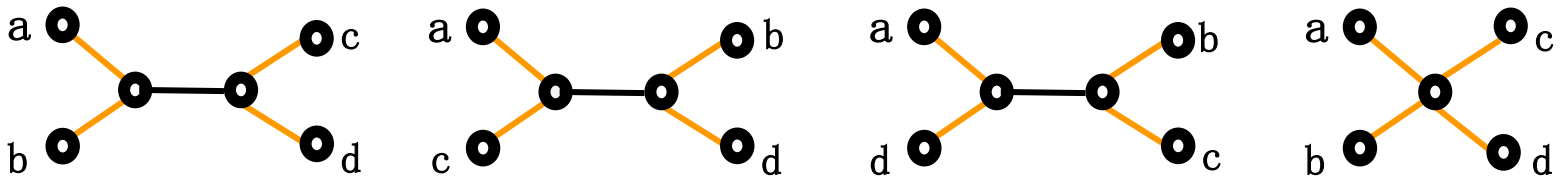
If there are four labels a, b, c, d for which the quartets in F_1 and F_2 are different, then one of the edges in the quartet in F_1 is not in any MAF of F_1 and F_2.

[Shi-Wang-Yang-Feng-Li-JC 2016] **An $O^*(4^k)$-time algorithm for the MAF problem on two unrooted multifurcating trees.**

Two General X-trees (quick overview)

Branching based on global inconsistency

quartet on four labels a, b, c, d



Observation.

If there are four labels a, b, c, d for which the quartets in F_1 and F_2 are different, then one of the edges in the quartet in F_1 is not in any MAF of F_1 and F_2.

[Shi-Wang-Yang-Feng-Li-JC 2016] **An $O^*(4^k)$-time algorithm for the MAF problem on two unrooted multifurcating trees.**

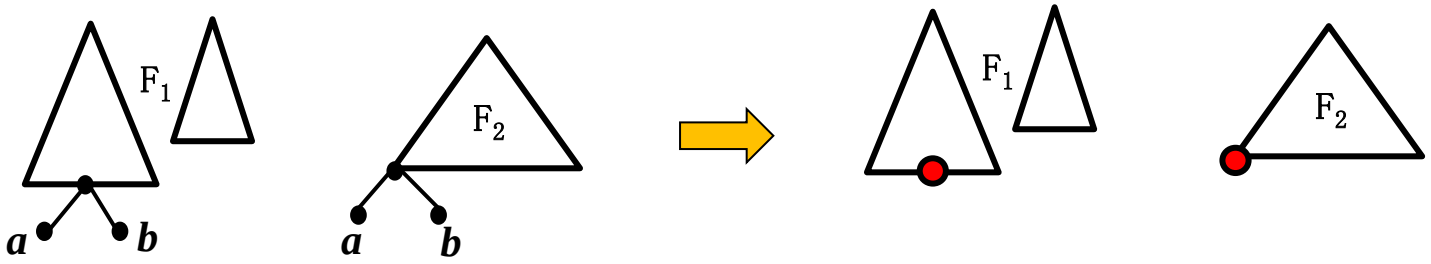
Remark. **Hallett-McCartin initiated this method for MAF on two unrooted binary trees.** [Hallett-McCartin 2007]

Two General X-trees (quick overview)

Branching based on local inconsistency

Review the process on binary trees

1. **Pick a sibling pair *a* and *b* in F_2;**
2. **If *a* and *b* are also siblings in F_1: shrink them;**

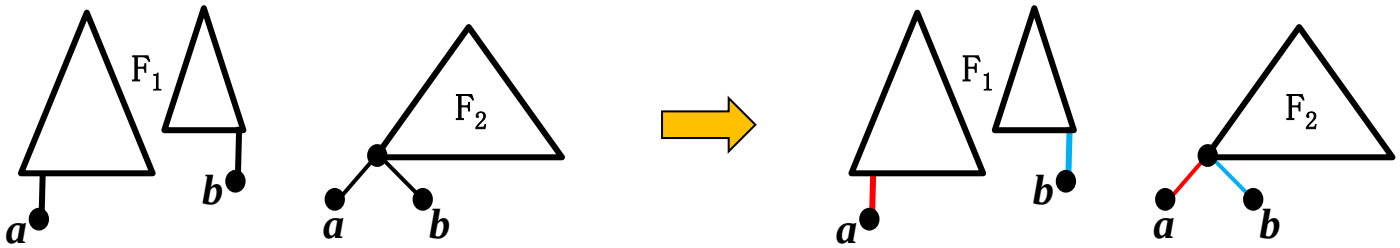


Two General X-trees (quick overview)

Branching based on local inconsistency

Review the process on binary trees

1. **Pick a sibling pair *a* and *b* in F_2;**
2. **If *a* and *b* are also siblings in F_1: shrink them;**
3. **If *a* and *b* are in different trees in F_1, branch on making either *a* or *b* as a single-vertex tree;**

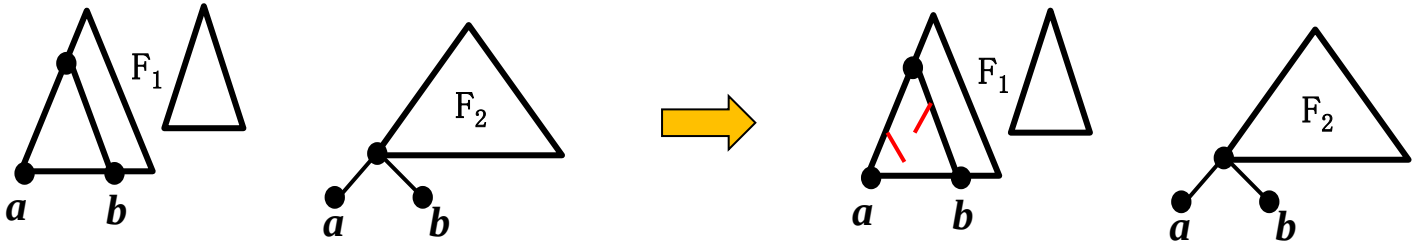


Two General X-trees (quick overview)

Branching based on local inconsistency

Review the process on binary trees

1. Pick a sibling pair a and b in F_2;
2. If a and b are also siblings in F_1: shrink them;
3. If a and b are in different trees in F_1, branch on making either a or b as a single-vertex tree;
4. Otherwise, branch on removing the edge incident to a, removing the edge incident to b, and removing all edges incident to the path connecting a and b.

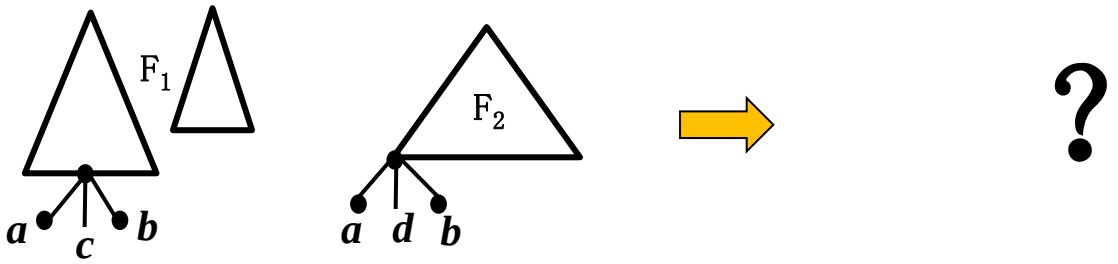


Two General X-trees (quick overview)

Branching based on local inconsistency

On multifurcating trees

1. Pick a sibling pair a and b in F_2;
2. If a and b are also siblings in F_1: shrink them;
3. If a and b are in different trees in F_1, branch on making either a or b as a single-vertex tree;
4. Otherwise, branch on removing the edge incident to a, removing the edge incident to b, and removing all edges incident to the path connecting a and b.

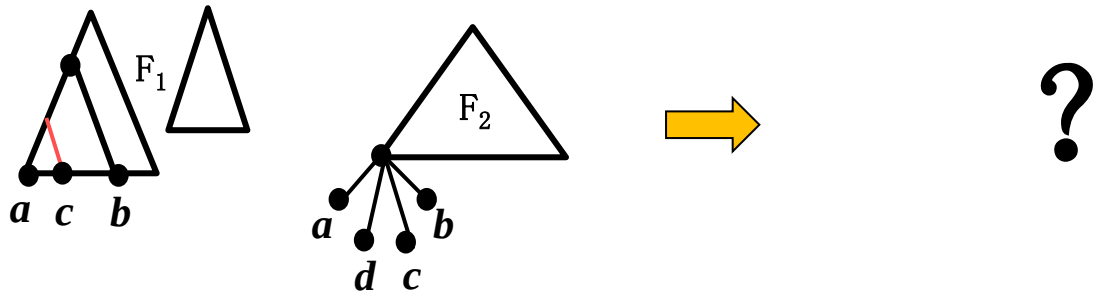


Two General X-trees (quick overview)

Branching based on local inconsistency

On multifurcating trees

1. Pick a sibling pair a and b in F_2;
2. If a and b are also siblings in F_1: shrink them;
3. If a and b are in different trees in F_1, branch on making either a or b as a single-vertex tree;
4. Otherwise, branch on removing the edge incident to a, removing the edge incident to b, and removing all edges incident to the path connecting a and b.



Two General X-trees (quick overview)

Branching based on local inconsistency

Thus, for general X-forest, we need new techniques to analyze the sibling structures.

We introduced a concept of BSS (basic sibling set), and apply branch-and-search based on BSS. For an inconsistency structure in BSS, we can identify at most 3 edges in an X-forest in which one must be removed.

Two General X-trees (quick overview)

Branching based on local inconsistency

Thus, for general X-forest, we need new techniques to analyze the sibling structures.

We introduced a concept of BSS (basic sibling set), and apply branch-and-search based on BSS. For an inconsistency structure in BSS, we can identify at most 3 edges in an X-forest in which one must be removed.

This gives an $O^*(3^k)$-time parameterized algorithm for two unrooted multifurcating trees [JC-Fan-Sze 2013]:

Two General X-trees (quick overview)

Branching based on local inconsistency

Thus, for general X-forest, we need new techniques to analyze the sibling structures.

We introduced a concept of BSS (basic sibling set), and apply branch-and-search based on BSS. For an inconsistency structure in BSS, we can identify at most 3 edges in an X-forest in which one must be removed.

This gives an $O^*(3^k)$-time parameterized algorithm for two unrooted multifurcating trees [JC-Fan-Sze 2013]:

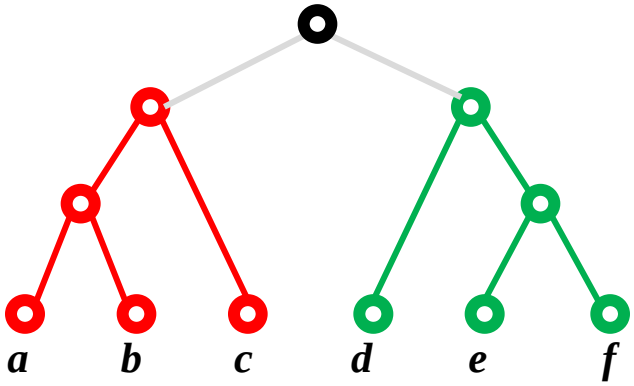
[Whidden-Beiko-Zeh 2016] developed an algorithm of time O*(2.42k) for two rooted multifurcating trees, assuming soft multifurcations

Two General X-trees (quick overview)

This looks rather straightforward.

What is difficult/tricky?

Not all edges are essential



Therefore, we need a very careful formulation of the problem (forest-structured sets are useful), and operations are only on “essential” edges.

Multiple X-trees

MAF on Multiple Trees:

Given a collection **C** = $\{T_1, T_2, \ldots, T_h\}$ of X-trees over the same label-set X, and a parameter k, is there an agreement forest of order at most k for the collection?

Multiple X-trees

MAF on Multiple Forests:
Given a collection **C** = {F_1, F_2, …, F_h} of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

Multiple X-trees

What is difficult on multiple trees?

Multiple X-trees

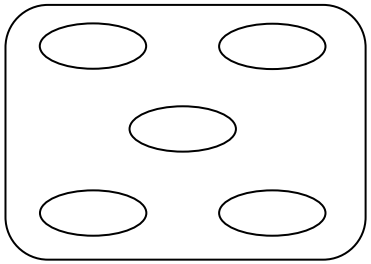
What is difficult on multiple trees?

- Consider an instance $C = \{F_1, F_2, \ldots, F_h\}$
- Increase the order of an AF may require to delete many edges in *C* (thus many branches);

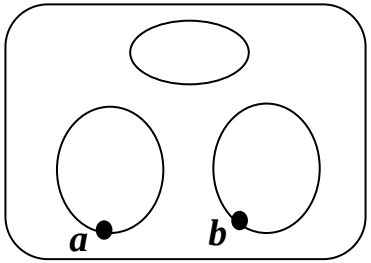
Multiple X-trees

What is difficult on multiple trees?

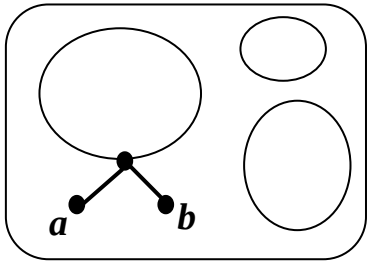
- Consider an instance $C = \{F_1, F_2, \ldots, F_h\}$
- Increase the order of an AF may require to delete many edges in C (thus many branches);
- Branching on a forest of small order “wastes” time (without increase the order of an AF).



F_1



F_2

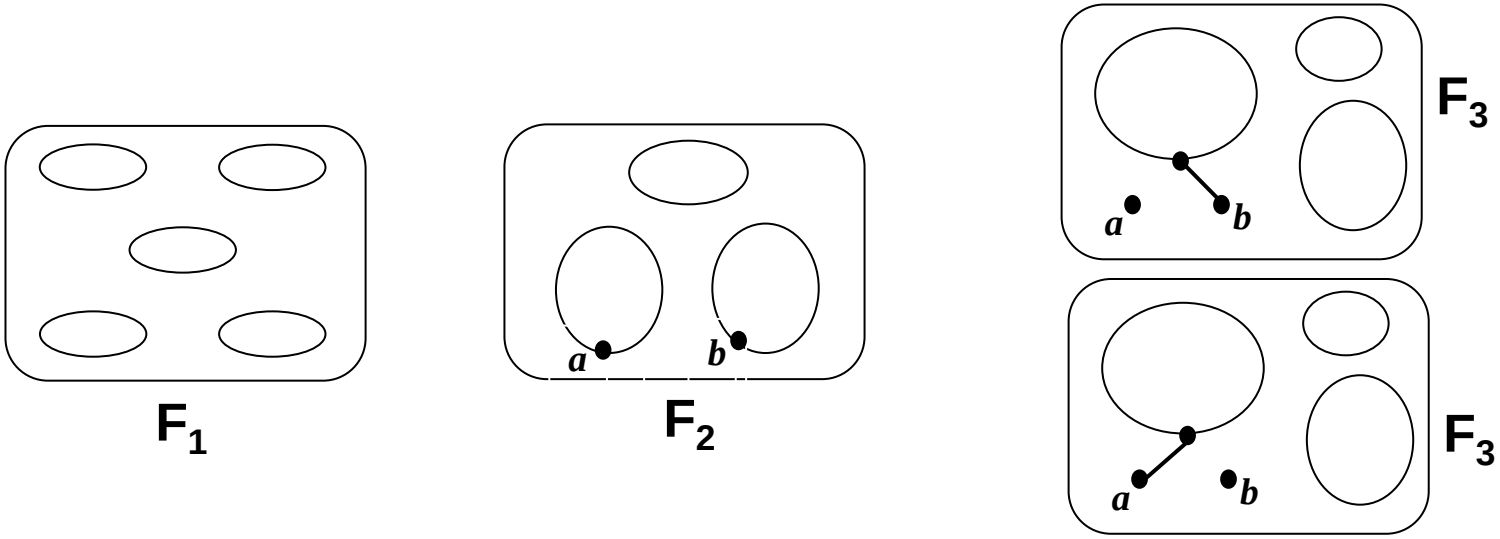


F_3

Multiple X-trees

What is difficult on multiple trees?

- Consider an instance $C = \{F_1, F_2, \ldots, F_h\}$
- Increase the order of an AF may require to delete many edges in *C* (thus many branches);
- Branching on a forest of small order “wastes” time (without increase the order of an AF).



Multiple X-trees

MAF on Multiple Forests:

Given a collection **C** = $\{F_1, F_2, \ldots, F_h\}$ of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

Multiple X-trees

MAF on Multiple Forests:
Given a collection **C** = $\{F_1, F_2, \ldots, F_h\}$ of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

Observation. A MAF for the collection C is an agreement forest for F_1 and F_2

Multiple X-trees

MAF on Multiple Forests:
Given a collection **C** = $\{F_1, F_2, \ldots, F_h\}$ of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

Observation. A MAF for the collection C is an agreement forest for F_1 and F_2

Basic idea. Examine all agreement forests for F_1 and F_2, and check if each of them is an MAF for the collection C

Multiple X-trees

MAF on Multiple Forests:

Given a collection **C** = {F_1, F_2, …, F_h} of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

For a collection C of rooted binary trees:

Using inconsistency in F_1 and F_2, identify no more than 3 edges in F_1, in which one of the edges must be removed.

An $O^*(3^k)$-time algorithm for the problem [Shi-Wang-JC 2013]

Multiple X-trees

MAF on Multiple Forests:

Given a collection **C** = $\{F_1, F_2, \ldots, F_h\}$ of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

For a collection C of rooted binary trees:

Using inconsistency in F_1 and F_2, identify no more than 3 edges in F_1, in which one of the edges must be removed.

An $O^*(3^k)$-time algorithm for the problem [Shi-Wang-JC 2013]

***[Z.Chen-Wang 2012] developed an $O^*(6^k)$-time algorithm for the problem.**

Multiple X-trees

MAF on Multiple Forests:

Given a collection **C** = $\{F_1, F_2, \ldots, F_h\}$ of X-forests over the same label-set X, and a parameter k, is there an agreement forest whose order is at most k larger than the largest forest order in the collection?

For a collection C of rooted binary trees:

Using inconsistency in F_1 and F_2, identify no more than 3 edges in F_1, in which one of the edges must be removed.

An $O^*(3^k)$-time algorithm for the problem [Shi-Wang-JC 2013]

***[Z.Chen-Wang 2012] developed an $O^*(6^k)$-time algorithm for the problem.**

For a collection C of unrooted binary trees:

Using inconsistency in F_1 and F_2, identify no more than 4 edges in F_1, in which one of the edges must be removed.

An $O^*(4^k)$-time algorithm for the problem [Shi-Wang-JC 2013]

Multiple X-trees

In principle, the above techniques can be used for developing parameterized algorithms for multiple general X-trees.

However, the multifurcating structures cause much more tedious case-by-case analysis, and also make the branch-and-search process much less efficient.

We would look for conceptually simpler and more efficient methods.

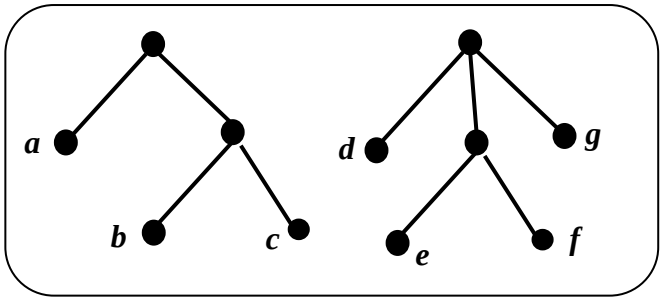
Multiple X-trees

Recall that one source that causes inefficiency is sometimes we have to branch on a forest of small order.

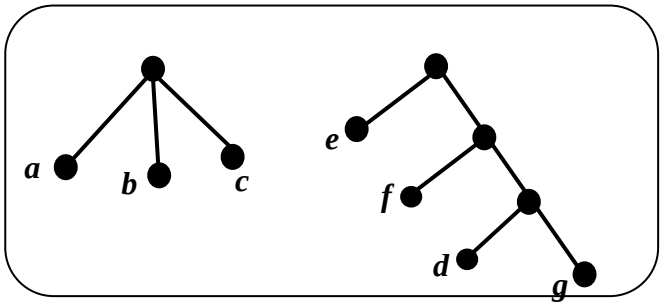
Thus, if we can maintain the condition that all forests in the collection are of the same order, then this undesired situation will be avoided.

Multiple X-trees

- Two X-forests are label isomorphic if they have the same label partitions (the corresponding trees may not necessarily be isomorphic).
- A collection $C = \{F_1, F_2, \ldots, F_h\}$ is label isomorphic if every pair of forests in the collection are label isomorphic.



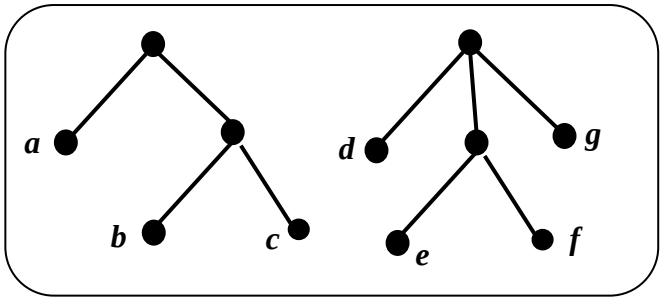
F_1



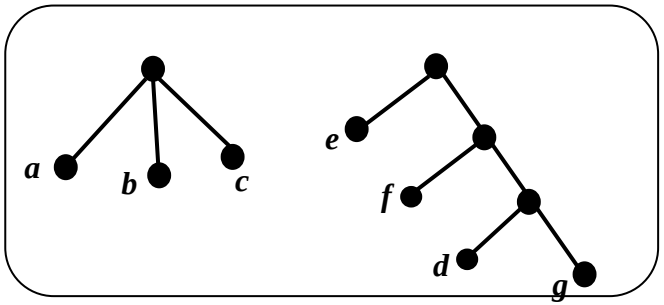
F_2

Multiple X-trees

- Two X-forests are label isomorphic if they have the same label partitions (the corresponding trees may not necessarily be isomorphic).
- A collection $C = \{F_1, F_2, \ldots, F_h\}$ is label isomorphic if every pair of forests in the collection are label isomorphic.



F_1

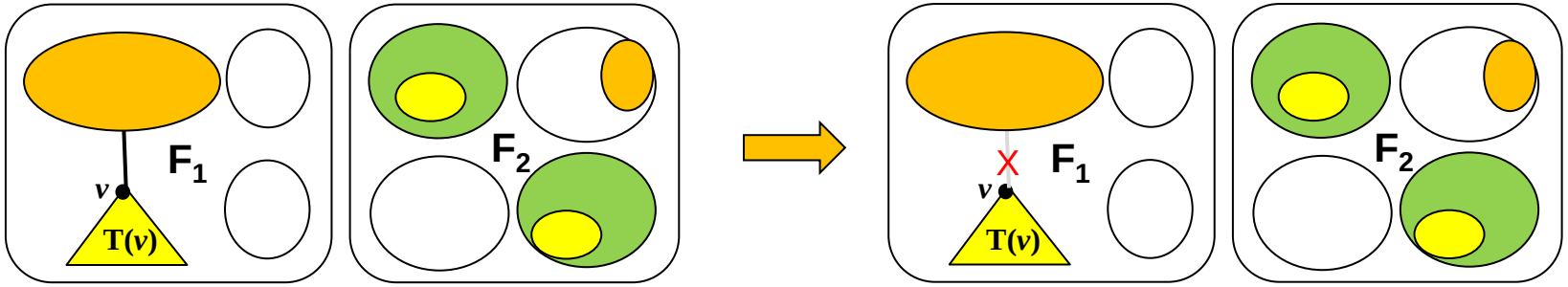


F_2

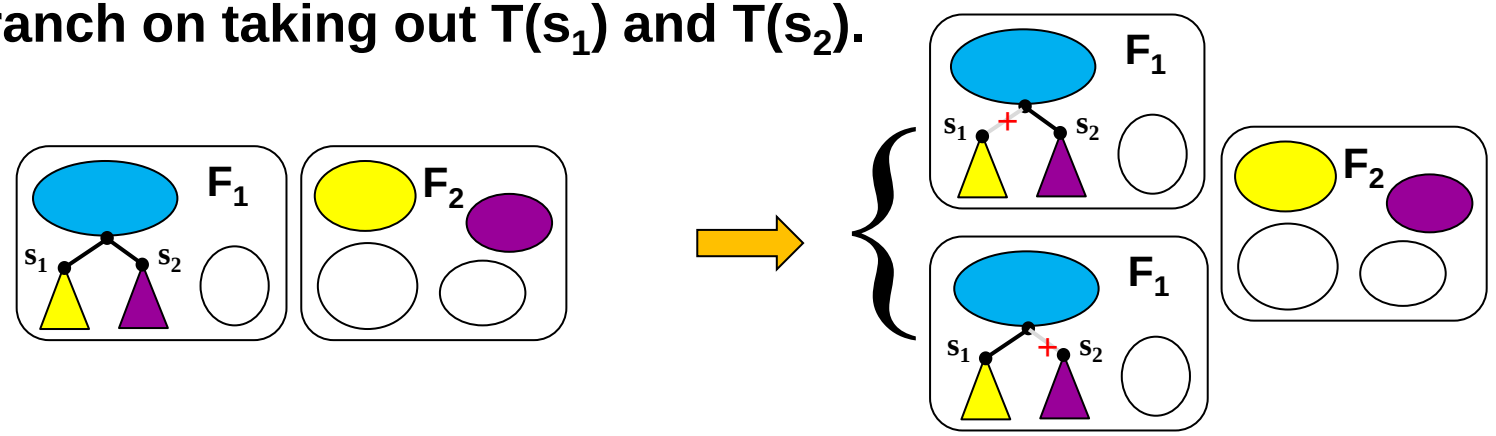
How expensive is it to achieve label isomorphism?

Achieving Label Isomorphism

Reduction Rule 1. If F_1 has a vertex v and F_2 has trees T_1, …, T_d such that $L(v) = T \cap (T_1 \cup \ldots \cup T_d)$, then take T(v) out of T.



Branching Rule 1. If F_1 has two siblings s_1 and s_2 and F_2 has trees T_1, T_2 such that $L(s_1) \subseteq L(T_1)$ and $L(s_2) \subseteq L(T_2)$, then branch on taking out $T(s_1)$ and $T(s_2)$.



Achieving Label Isomorphism

Theorm. **If neither Reduction Rule 1 nor Branching Rule 1 is applicable on F_1 and F_2, then F_1 and F_2 are label-isomorphic.**

Achieving Label Isomorphism

Theorm. **If neither Reduction Rule 1 nor Branching Rule 1 is applicable on F_1 and F_2, then F_1 and F_2 are label-isomorphic.**

How expensive is Branching Rule 1?

Achieving Label Isomorphism

Theorm. **If neither Reduction Rule 1 nor Branching Rule 1 is applicable on F_1 and F_2, then F_1 and F_2 are label-isomorphic.**

How expensive is Branching Rule 1?

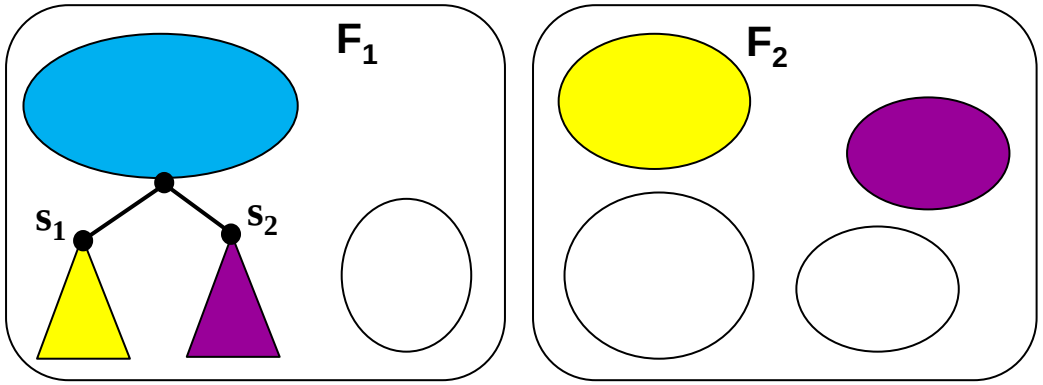
Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

Achieving Label Isomorphism

Theorm. **If neither Reduction Rule 1 nor Branching Rule 1 is applicable on F_1 and F_2, then F_1 and F_2 are label-isomorphic.**

How expensive is Branching Rule 1?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, $a', b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

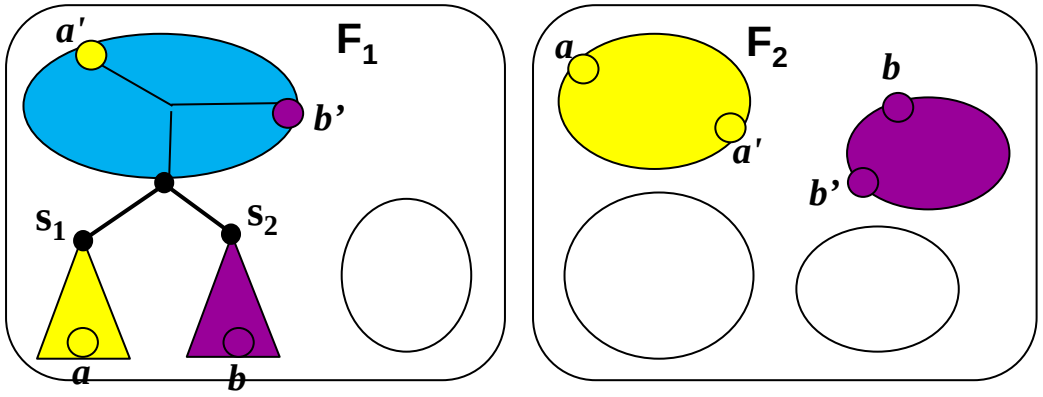


Achieving Label Isomorphism

Theorm. **If neither Reduction Rule 1 nor Branching Rule 1 is applicable on F_1 and F_2, then F_1 and F_2 are label-isomorphic.**

How expensive is Branching Rule 1?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**



Why does this lemma help?

Lemma. When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.

Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule is needed to make F_1 and F_2 label isomorphic

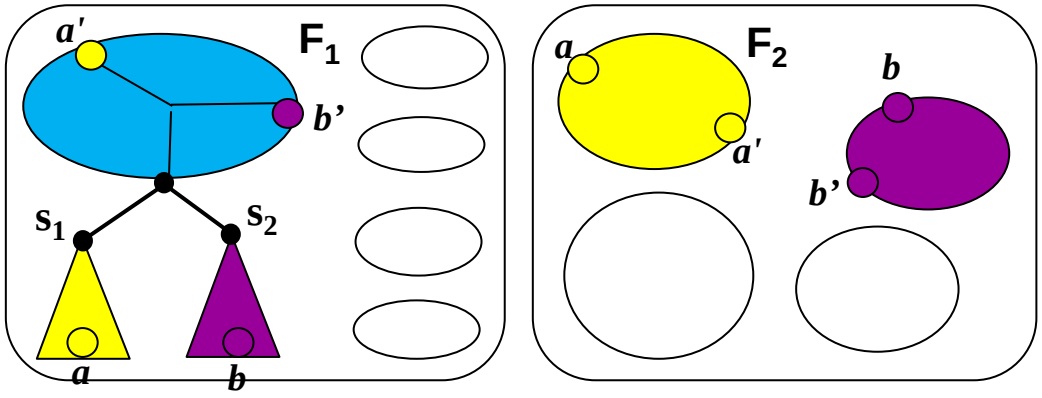
If $Ord(F_1) \geq Ord(F_2)$

Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) \geq Ord(F_2)$

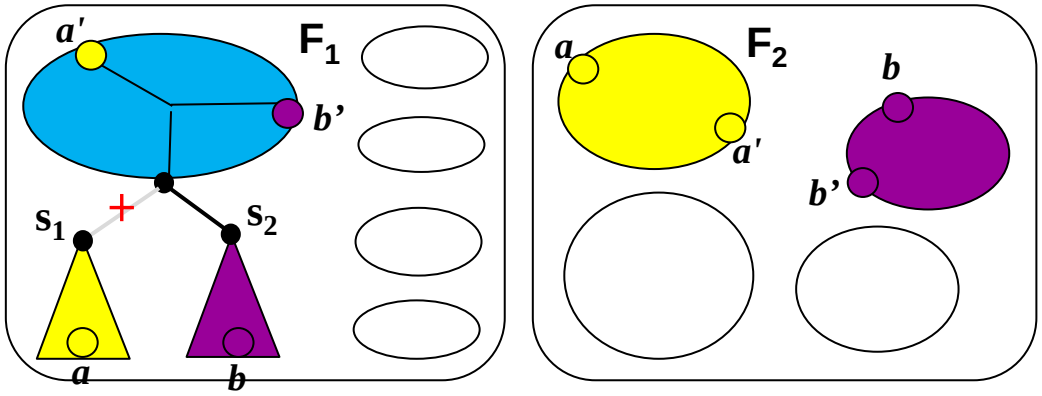


Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) \geq Ord(F_2)$



Branching Rule 1 branches on $L(s_1)$

Why does this lemma help?

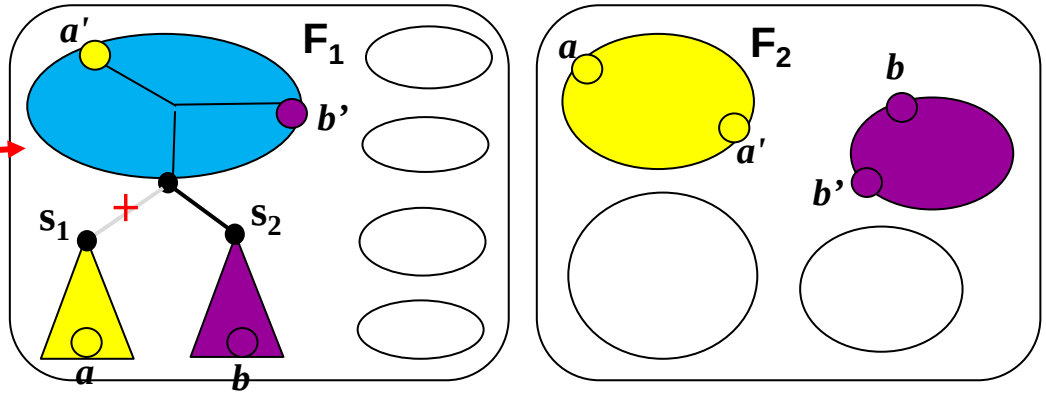
Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) \geq Ord(F_2)$

this tree contains labels in different trees in F_2

Branching Rule 1 branches on $L(s_1)$



Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

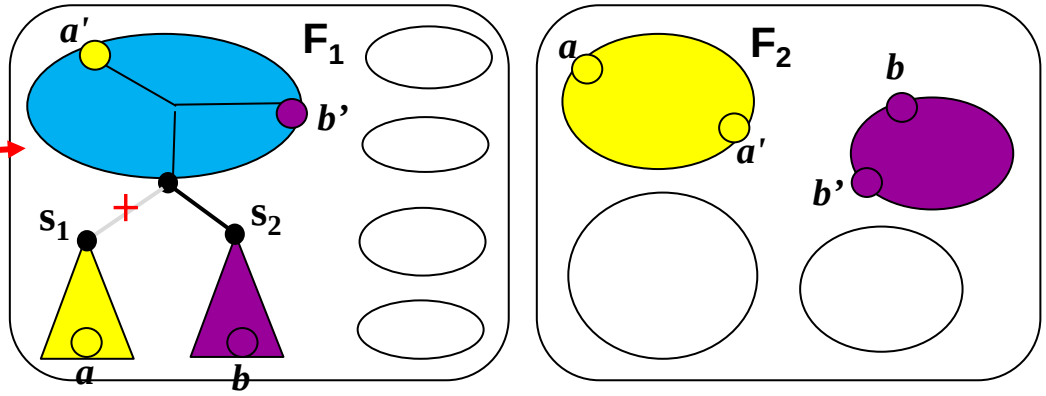
To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) \geq Ord(F_2)$

Thus Reduction Rule 1 will further split it

this tree contains labels in different trees in F_2

Branching Rule 1 branches on $L(s_1)$



Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

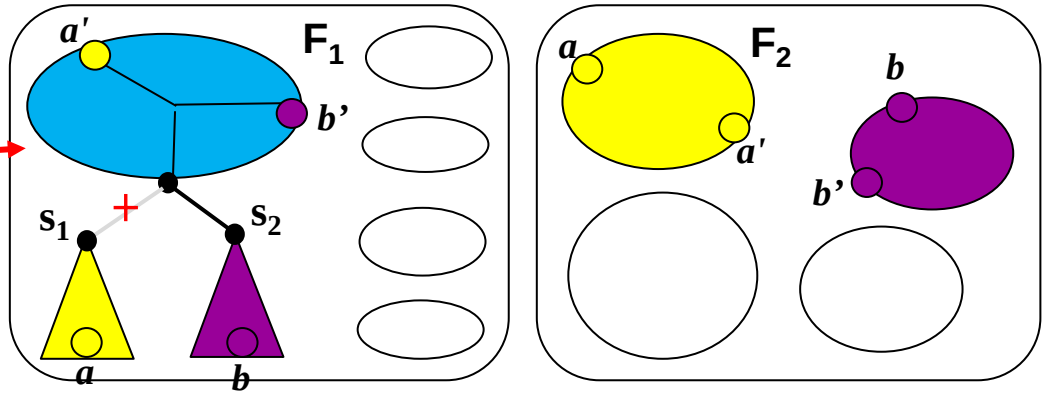
To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) \geq Ord(F_2)$

Thus Reduction Rule 1 will further split it

this tree contains labels in different trees in F_2

Branching Rule 1 branches on $L(s_1)$



Thus, an application of Branching Rule 1 on the forest of larger order will decrease the parameter by at least 2

Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

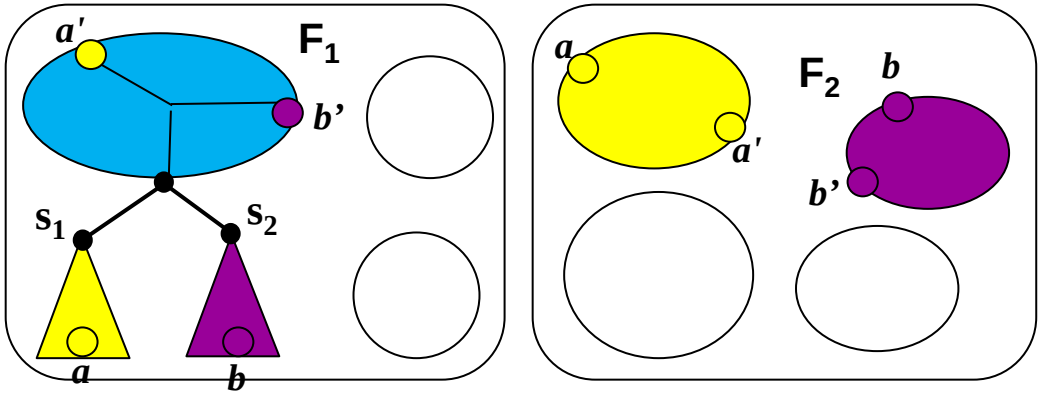
If $Ord(F_1) < Ord(F_2)$

Why does this lemma help?

Lemma. When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) < Ord(F_2)$

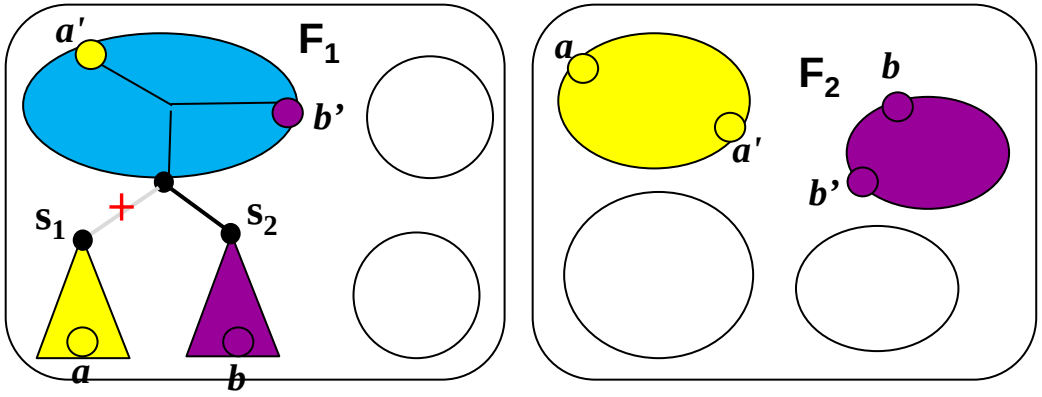


Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) < Ord(F_2)$



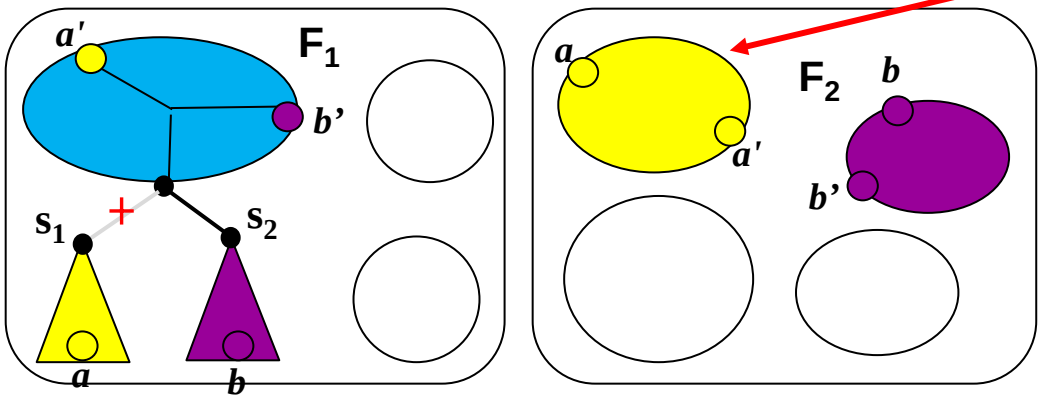
Branching Rule 1 branches on $L(s_1)$

Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) < Ord(F_2)$



this tree contains labels in different trees in F_1

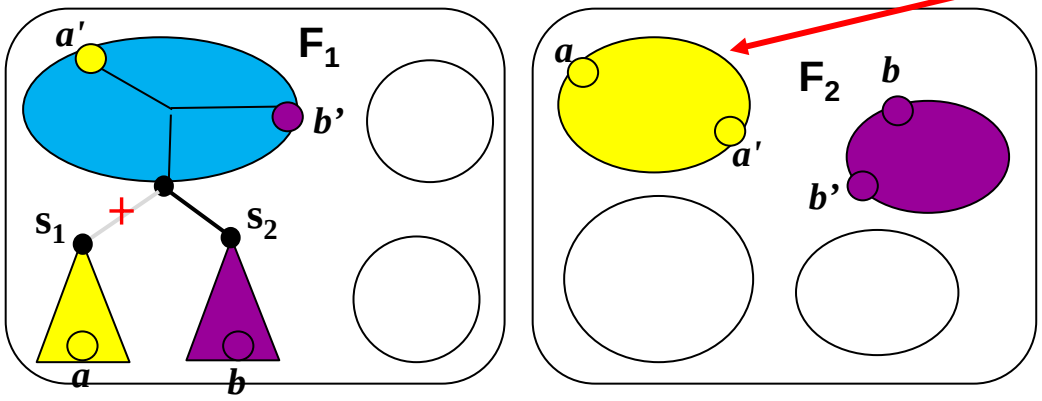
Branching Rule 1 branches on $L(s_1)$

Why does this lemma help?

Lemma. When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) < Ord(F_2)$



this tree contains labels in different trees in F_1

Thus Reduction Rule 1 will further split it

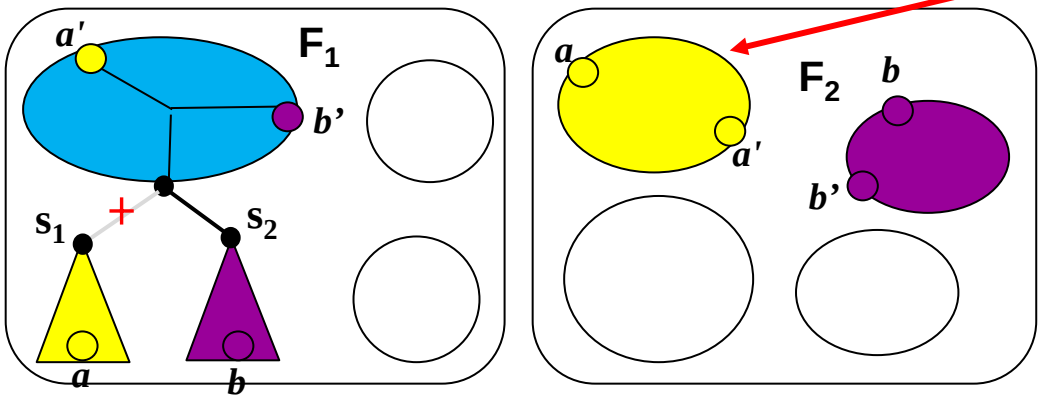
Branching Rule 1 branches on $L(s_1)$

Why does this lemma help?

Lemma. **When Branching Rule 1 is applicable on siblings s_1 and s_2 in a tree T in F_1 based on F_2, there are labels a, a', b, b' in T, with $a \in L(s_1)$, $b \in L(s_2)$, a', $b' \notin L(s_1) \cup L(s_2)$, such that a and a' (resp. b and b') are in the same tree in F_2.**

To get a feeling, assume that only one Reduction Rule Is needed to make F_1 and F_2 label isomorphic

If $Ord(F_1) < Ord(F_2)$



this tree contains labels in different trees in F_1

Thus Reduction Rule 1 will further split it

Branching Rule 1 branches on $L(s_1)$

Thus, an application of Branching Rule 1 on the forest of smaller order will also decrease the parameter by at least 1

Achieving Label Isomorphism

Theorem. Let Φ_1 and Φ_2 be label-isomorphic X-forests, obtained by h applications of Branching Rule 1 on X-forests F_1 and F_2. Then

$$Ord(\Phi_1) = Ord(\Phi_2) \geq h + \max\{Ord(F_1), Ord(F_2)\}$$

Achieving Label Isomorphism

Theorem. Let Φ_1 and Φ_2 be label-isomorphic X-forests, obtained by h applications of Branching Rule 1 on X-forests F_1 and F_2. Then

$$Ord(\Phi_1) = Ord(\Phi_2) \geq h + \max\{Ord(F_1), Ord(F_2)\}$$

Thus, Reduction Rule 1 and Branching Rule 1 not only make label isomorphism of X-forests, but also effectively decrease the parameter value k — at least as good as the recurrence relation T(k) = 2 T(k-1)

Achieving Label Isomorphism

Theorem. Let Φ_1 and Φ_2 be label-isomorphic X-forests, obtained by h applications of Branching Rule 1 on X-forests F_1 and F_2. Then

$$Ord(\Phi_1) = Ord(\Phi_2) \geq h + \max\{Ord(F_1), Ord(F_2)\}$$

Thus, Reduction Rule 1 and Branching Rule 1 not only make label isomorphism of X-forests, but also effectively decrease the parameter value k — at least as good as the recurrence relation T(k) = 2 T(k-1)

For a given instance $\{F_1, F_2, \ldots, F_h\}$, we always pick an F_p of the largest order and an F_q that is not label-isomorphic to F_p, and make them label-isomorphic. This process decreases the parameter k at least as good as T(k) = 2 T(k-1)

Multiple X-trees

Observation. **Any MAF algorithm of running time $O^*(c^k)$ with $c \geq 2$ on two X-forests, based on exhaustive search can be translated into an $O^*(c^k)$-time MAF algorithm on multiple X-forests.**

Remark. **This seems to provide a rather general tool to reduce the MAF problem on multiple trees to the problem with two trees.**

Corollary. **The MAF problem on multiple rooted general X-trees can be solved in time $O^*(2.42^k)$.**

Approximation Algorithms

Approximation Algorithms

Earlier Work:

- **Ratio-3 for 2 rooted binary trees [Hein et al. 1996]**

Approximation Algorithms

Earlier Work:

- Ratio-3 for 2 rooted binary trees [Hein et al. 1996]
- The algorithm of Hein et al has ratio ≥ 4, and a new ratio-3 algorithm [Rodrigues et al. 2001]

Approximation Algorithms

Earlier Work:

- Ratio-3 for 2 rooted binary trees [Hein et al. 1996]
- The algorithm of Hein et al has ratio ≥ 4, and a new ratio-3 algorithm [Rodrigues et al. 2001]
- The algorithm of Hein et al. has ratio ≥ 5, and the algorithm of Rodrigues et al. has ratio ≥ 4 [Bonet et al. 2006]

Approximation Algorithms

Earlier Work:

- Ratio-3 for 2 rooted binary trees [Hein et al. 1996]
- The algorithm of Hein et al has ratio ≥ 4, and a new ratio-3 algorithm [Rodrigues et al. 2001]
- The algorithm of Hein et al. has ratio ≥ 5, and the algorithm of Rodrigues et al. has ratio ≥ 4 [Bonet et al. 2006]
- Ratio-3 for 2 rooted binary trees [Rodrigues et al. 2007]
- Ratio-3 for 2 rooted binary trees [Bordewich et al. 2008]

Approximation Algorithms

Earlier Work:

- Ratio-3 for 2 rooted binary trees [Hein et al. 1996]
- The algorithm of Hein et al has ratio ≥ 4, and a new ratio-3 algorithm [Rodrigues et al. 2001]
- The algorithm of Hein et al. has ratio ≥ 5, and the algorithm of Rodrigues et al. has ratio ≥ 4 [Bonet et al. 2006]
- Ratio-3 for 2 rooted binary trees [Rodrigues et al. 2007]
- Ratio-3 for 2 rooted binary trees [Bordewich et al. 2008]
- Ratio-(d+1) for two rooted general trees [Rodrigues et al. 2007]
- Ratio-8 for multiple rooted binary trees [Chataigner 2005]

Approximation Algorithms

More recent work on two binary trees

For two rooted binary trees

- **3-approximation in linear time [Whidden-Zeh 2009]**
- **2.5-approximation [Shi et al. 2016]**
- **2-approximation [Schalekamp-Zuylen-Ster 2016] (using LP Duality)**

Approximation Algorithms

More recent work on two binary trees

For two rooted binary trees

- **3-approximation in linear time [Whidden-Zeh 2009]**
- **2.5-approximation [Shi et al. 2016]**
- **2-approximation [Schalekamp-Zuylen-Ster 2016] (using LP Duality)**

For two unrooted binary trees

- **3-approximation [Whidden-Zeh 2009]**
- **3-approximation [JC-Fan-Sze 2015]**

Approximation Algorithms

Our Approach

- **Basic idea:**

  In the study of parameterized algorithms for the MAF problem, our basic operation is to find a collection B of essential edges in which one must be removed, and branch on removing each of the edges in B

Approximation Algorithms

Our Approach

- **Basic idea:**

  In the study of parameterized algorithms for the MAF problem, our basic operation is to find a collection B of essential edges in which one must be removed, and branch on removing each of the edges in B

- **Thus:**

  If we remove all edges in B, then we get an approximation algorithm of ratio |B| for the MAF problem.

Approximation Algorithms

- **3-approximation for MAF on 2 unrooted general X-trees [JC-Fan-Sze 2013]**

  Approximations on 2 rooted (soft) general X-trees with ratio 4 [van Iersel et al. 2014] and 3 [Whidden et al. 2016]

- **3-approximation for MAF on multiple rooted binary trees [JC-Shi-Feng-Wang 2014]**

  3-approximation for the problem was independently developed by [Mukhopadhyay-Bhabak 2016]

- **4-approximation for MAF on multiple unrooted binary trees [JC-Shi-Feng-Wang 2014]**

Conclusions and Final Remarks

- MAF and related problems from evolutionary biology offer nice combinatorial structures for algorithmic research;
- The research is still in a fairly preliminary stage;
- Most developed techniques are elementary based on straightforward combinatorial structural analysis;
- Deeper insight and new techniques for more efficient algorithms?
- Lower bounds?
- Problem kernelizations?