ON THE WEIGHTED QUARTET CONSENSUS PROBLEM

Manuel Lafond[1]

Celine Scornavacca[2]

1 University of Ottawa, Canada

2 Institut des sciences de l'evolution de l'Université de Montpellier, France

The plan

1. What is a quartet ?

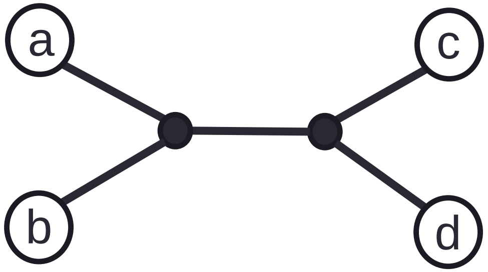
2. The weighted quartet consensus problem

3. NP-hardness

4. ‘Randomized’ 1/2 approximation algorithm

5. Derandomization

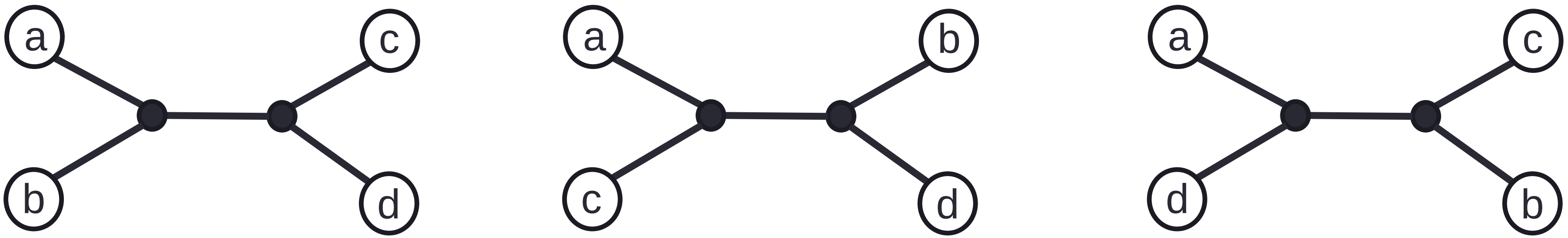
Quartet



Unrooted binary tree of four **labeled** leaves.

This quartet is denoted **ab|cd**.

Quartet

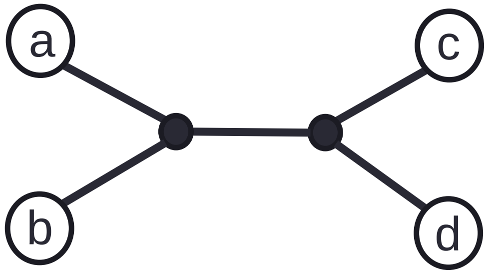


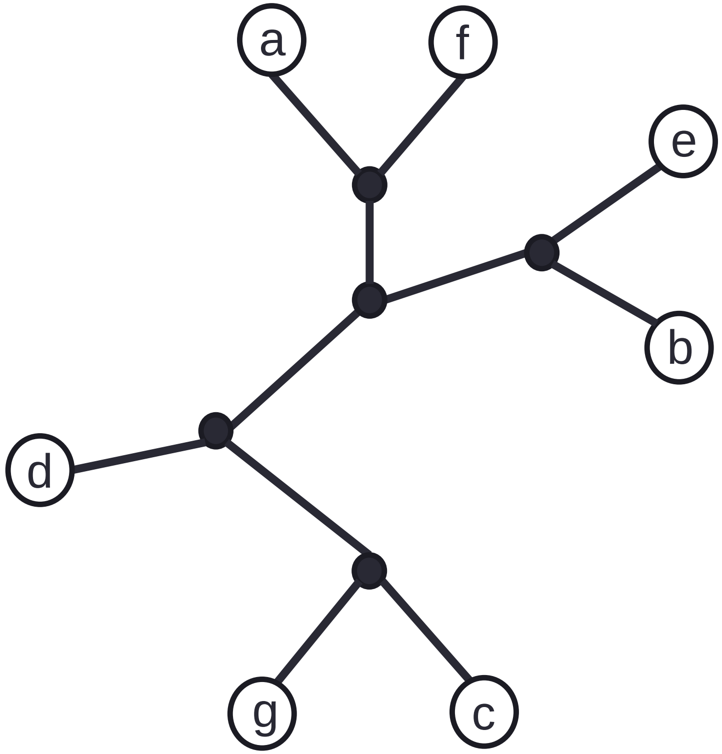
Three possible quartets on {a,b,c,d}:

ab|cd   ac|bd   ad|bc

Quartet in a tree

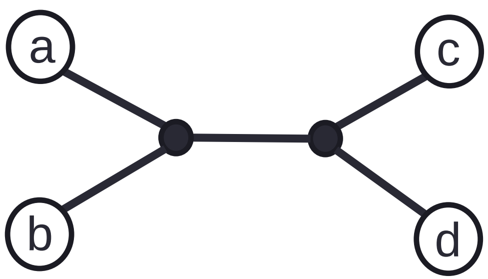
A tree **T contains ab|cd** if ab|cd is a subtree of T (in the minor sense, with labels preservation).
Equivalently, the **a-b path does not intersect the c-d** path (no shared vertex).

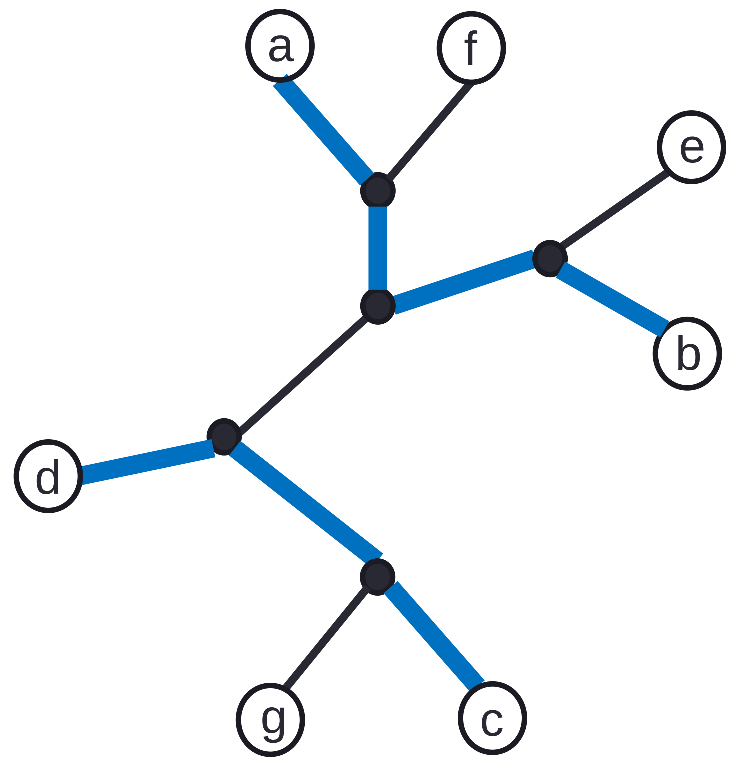




Quartet in a tree

A tree **T contains ab|cd** if ab|cd is a subtree of T (in the minor sense, with labels preservation).
Equivalently, the **a-b path does not intersect the c-d** path (no shared vertex).

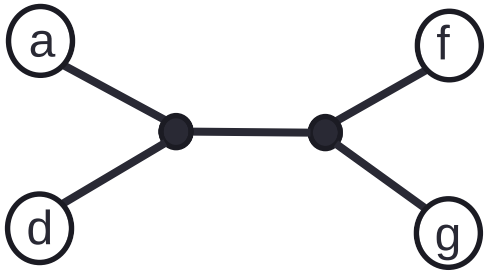


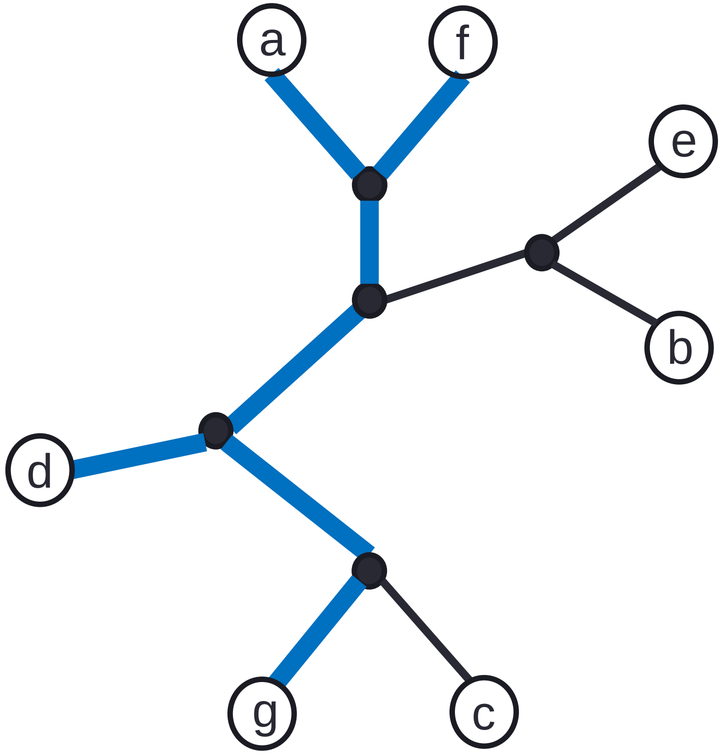


contains ab|cd

Quartet in a tree

A tree **T contains ab|cd** if ab|cd is a subtree of T (in the minor sense, with labels preservation).
Equivalently, the **a-b path does not intersect the c-d** path (no shared vertex).

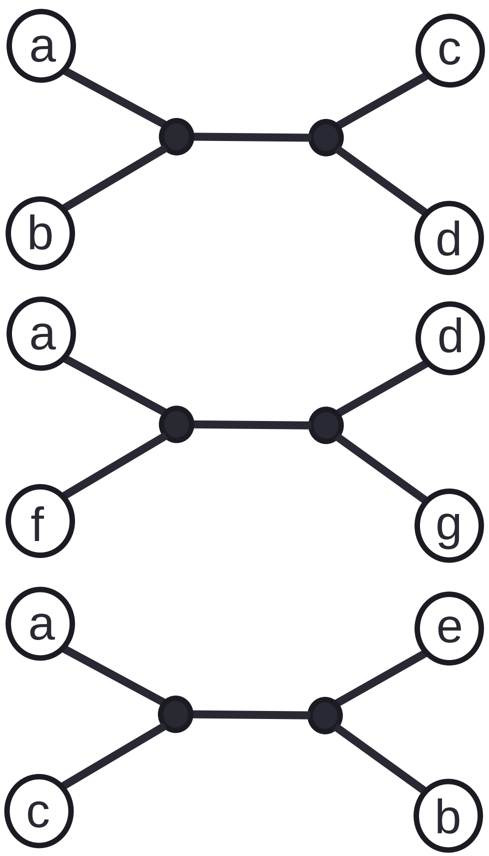




does not
contain ad|fg

Quartet consistency

Given: a set of quartets Q.
Task: find a tree T that contains every quartet in Q.

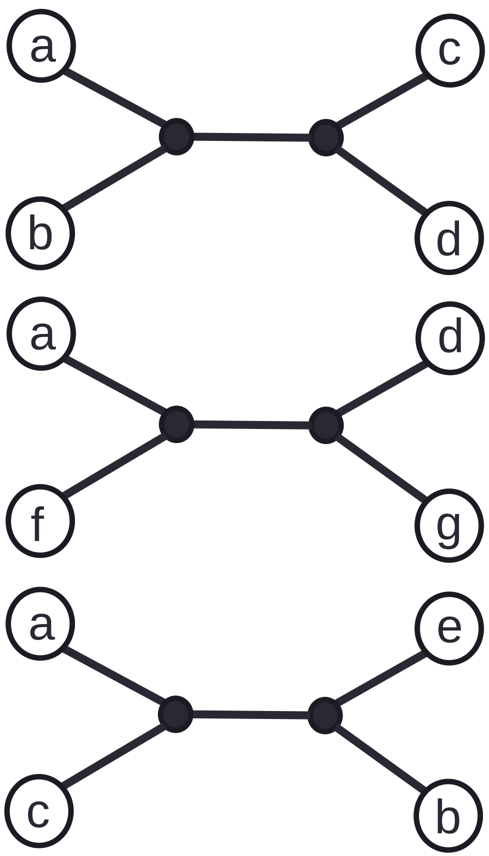


NP-hard [Steel, 1992]

?

Quartet consistency

Given: a set of quartets Q.
Task: find a tree T that contains **a maximum number of quartet** from Q.

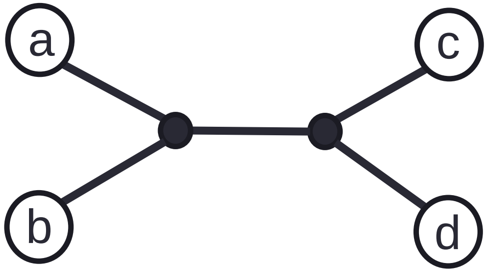


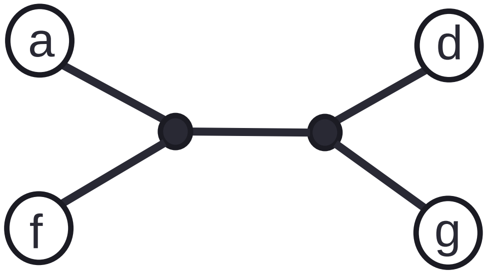
NP-hard [Steel, 1992]

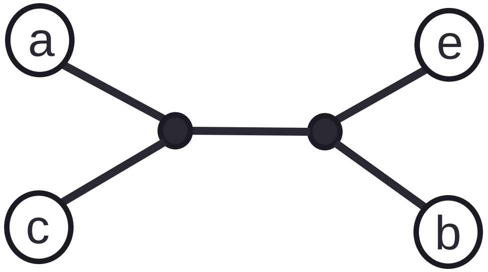
Max SNP-hard

?

Where do these quartets come from?

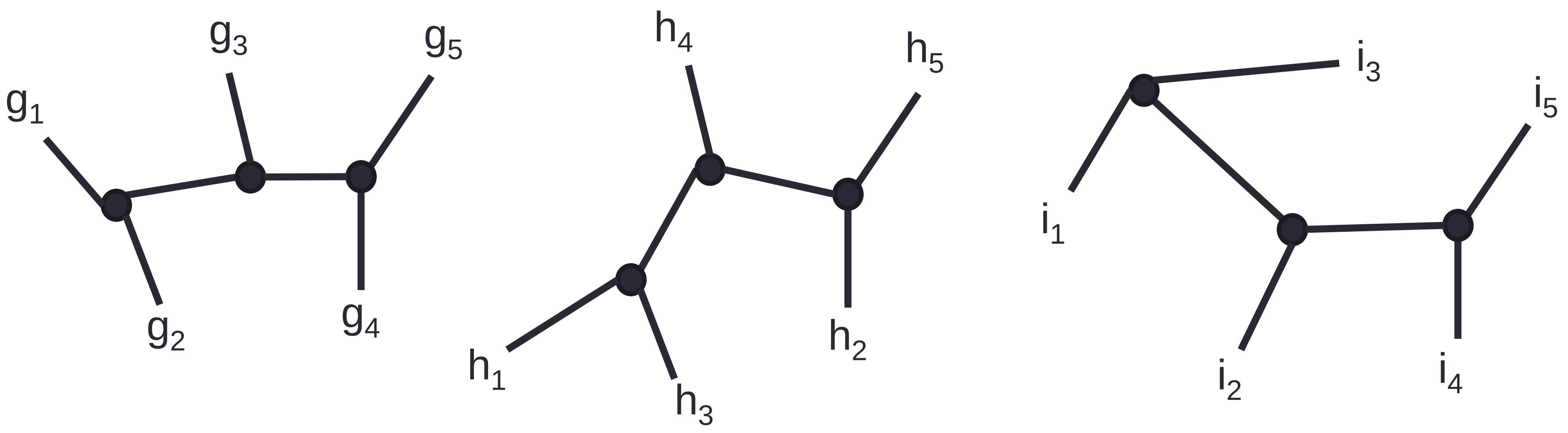






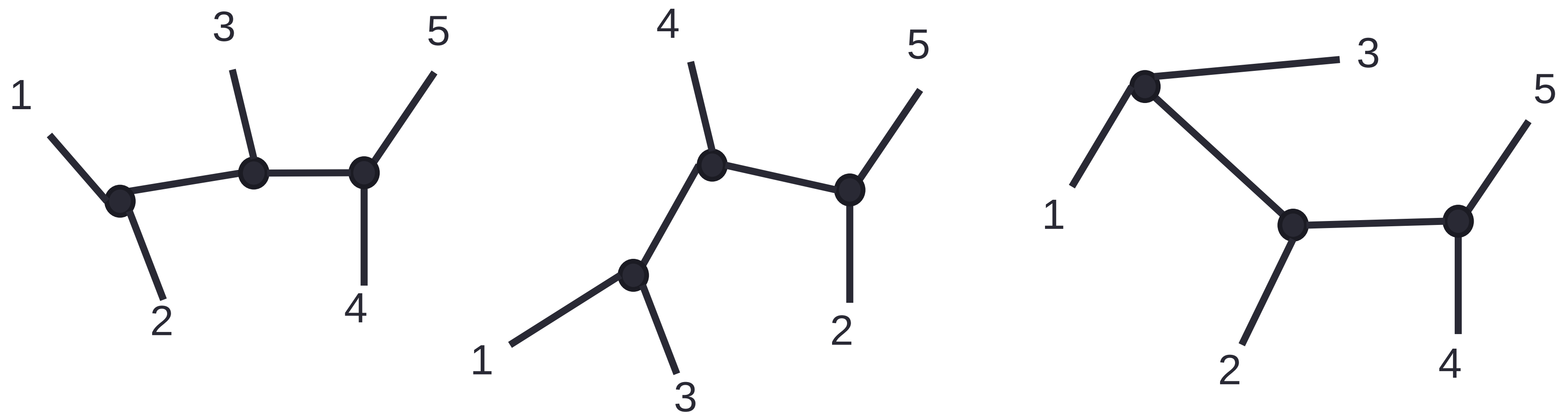
Where do these quartets come from?

Infer many gene trees (assumed to be binary).
The species tree should be a « consensus » of these trees
⇒ Replace the genes by the species that contains them.



Where do these quartets come from?

Infer many gene trees (assumed to be binary).
The species tree should be a « consensus » of these trees
⇒ Replace the genes by the species that contains them.



{1,2,3,4,5} is the set of species
We assume every tree is now on leafset {1,2,3,4,5}

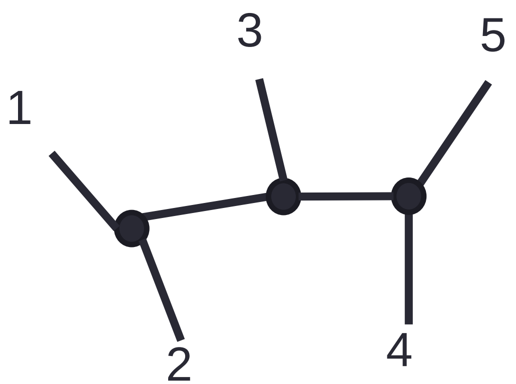
Now, combine these trees into a consensus.

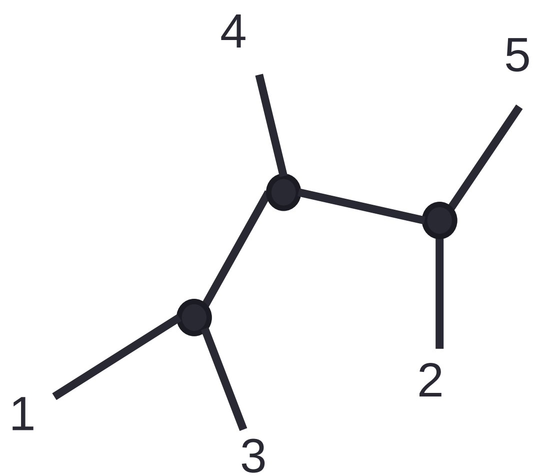
Where do these quartets come from?

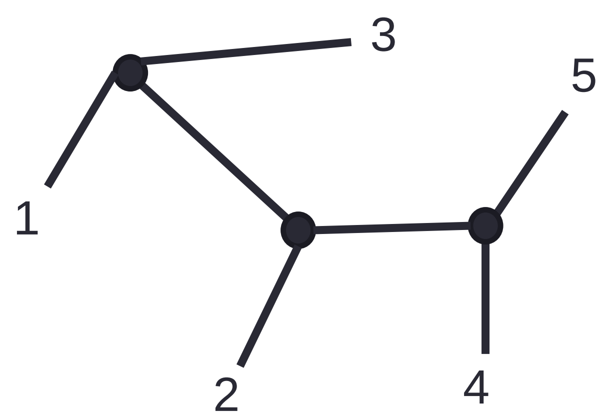
List each of the $\binom{n}{4}$ quartets contained in each tree

$\Rightarrow$ Multiset of quartets Q

Goal: find a tree that contains a **maximum number of quartets of Q** (in the multiset sense).







12|34	13|42	13|24
12|35	13|45	13|25
12|45	13|25	13|45
23|45	34|25	32|45

Where do these quartets come from?

List each of the $\binom{n}{4}$ quartets contained in each tree

$\Rightarrow$ Multiset of quartets Q

Goal: find a tree that contains a **maximum number of quartets of Q** (in the multiset sense).

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

?

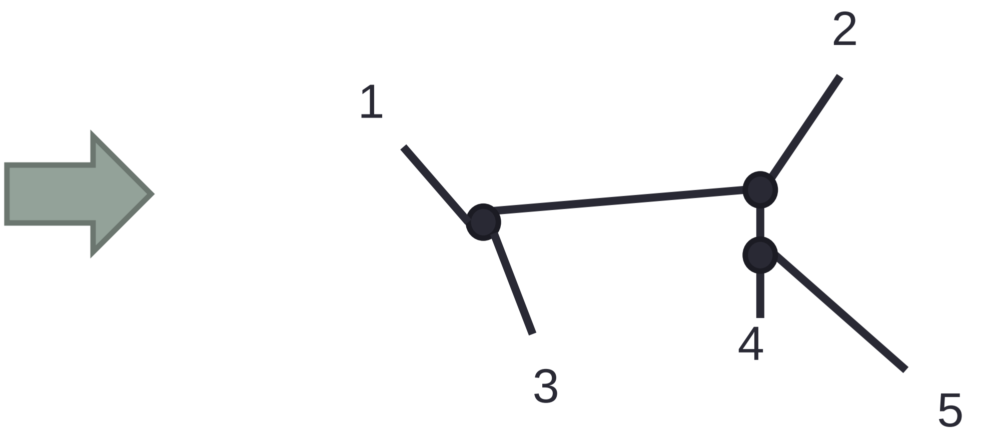
Where do these quartets come from?

List each of the $\binom{n}{4}$ quartets contained in each tree

$\Rightarrow$ Multiset of quartets Q

Goal: find a tree that contains a **maximum number of quartets of Q** (in the multiset sense).

~~12|34~~
~~12|35~~
12|45
23|45
13|42
13|45
13|25
~~34|25~~
13|24
13|25
13|45
32|45



Tree contains 9 quartets from Q (note that some quartets are counted twice)

Weighted quartet consensus (WQC)

Given: a set of trees $T_1, \ldots T_k$ on the same leafset.
Goal: find a tree that contains a **maximum number of quartets from**

quartets(T_1) $\uplus$ ... $\uplus$ quartets(T_k) $\uplus$ denotes multiset union

Weighted quartet consensus (WQC)

Given: a set of trees $T_1, \ldots T_k$ on the same leafset.
Goal: find a tree that contains a **maximum number of quartets from**

$$\textbf{quartets}(T_1) \uplus \ldots \uplus \textbf{quartets}(T_k)$$

$\uplus$ denotes multiset union

Why the name "weighted quartet consensus"?
Each quartet can be given a weight in [0..1] based on its frequency in the input trees.

e.g.

ab|cd	**0.24**
ac|bd	**0.47**
ad|bc	**0.29**

We want to find a tree that maximizes the sum of the weights of its quartets.

Some history

If input quartets **Q can be anything**

- NP-hard to find a tree with all quartets, Max SNP-hard to maximize [Steel, 1992]
- Many heuristics [Strimmet & von Haeseler, 1996, Berry et al., 1999, …]

Some history

If input quartets **Q can be anything**

- NP-hard to find a tree with all quartets, Max SNP-hard to maximize [Steel, 1992]
- Many heuristics [Strimmet & von Haeseler, 1996, Berry et al., 1999, …]

If input quartets **Q are dense**

- Meaning, Q has **exactly one quartet for each 4 labels**
- Easy to check if some tree contains all of Q
- NP-hard to maximize the number of quartets of Q in a tree [Jiang & al., 2001]
- Polynomial-time approximation scheme (PTAS) [Jiang & al., 2001]
- FPT in time $O(4^k n + n^4)$ [Gramm & Niedermeier, 2001], improved in [Chang, 2008]

Some history

If input quartets **Q can be anything**

- NP-hard to find a tree with all quartets, Max SNP-hard to maximize [Steel, 1992]
- Many heuristics [Strimmet & von Haeseler, 1996, Berry et al., 1999, …]

If input quartets **Q are dense**

- Meaning, Q has **exactly one quartet for each 4 labels**
- Easy to check if some tree contains all of Q
- NP-hard to maximize the number of quartets of Q in a tree [Jiang & al., 2001]
- Polynomial-time approximation scheme (PTAS) [Jiang & al., 2001]
- FPT in time $O(4^k n + n^4)$ [Gramm & Niedermeier, 2001], improved in [Chang, 2008]

If input quartets **Q come from trees on the same set of leaf labels (our setting)**

- Minimization version of WQC: minimize number of quartets to **discard from the multiset** to have a tree containing all remaining quartets [Bansa & al., 2011]
    - ***Conjectured that WQC is NP-hard***
    - 2-approximation: return the best input tree
- ASTRAL heuristic [Mirarab & al., 2014] (actually also a 2-approx.)
    - ***Conjectured that WQC is NP-hard***

In this work

- We resolve this conjecture by **proving that WQC is NP-hard.**

- We devise a 1/2-approximation algorithm

NP-hardness (idea)

Cyclic ordering problem

Given: a set S of n elements and a set C of ordered triples (a,b,c) of elements of S
Task: does there exist a linear ordering of S such that for each (a,b,c) ∈ C, we have either ***a* < *b* < *c*** or **b < c < a** or ***c* < *b* < *a***?

In other words, can the elements of S be arranged in a circle and contain every triple of C when read clockwise?

NP-hardness (idea)

Cyclic ordering problem

Given: a set S of n elements and a set C of ordered triples (a,b,c) of elements of S
Task: does there exist a linear ordering of S such that for each $(a,b,c) \in C$, we have either $\boldsymbol{a < b < c}$ or $\mathbf{b < c < a}$ or $\boldsymbol{c < b < a}$?

In other words, can the elements of S be arranged in a circle and contain every triple of C when read clockwise?

e.g.
(a, b, c)

(d, a, b)

(e, b, c)

(c, a, b)

a

e **b**

c **d**

NP-hardness (idea)

Cyclic ordering problem

Given: a set S of n elements and a set C of ordered triples (a,b,c) of elements of S
Task: does there exist a linear ordering of S such that for each (a,b,c) ∈ C, we have either ***a* < *b* < *c*** or **b < c < a** or ***c* < *b* < *a***?

In other words, can the elements of S be arranged in a circle and contain every triple of C when read clockwise?

NP-hard [Galil & Megiddo, 1977]

NP-hardness (idea)

Reduction: main gadget = W-Z trees

Every tree is on leafset W + Z + S,
where **W and Z are huge** (e.g. n^{100} leaves)
and $S = \{s_1, \ldots, s_n\}$

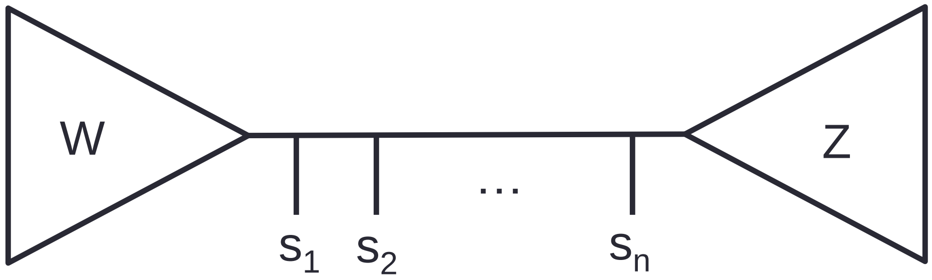
Each input tree has the s_i's linearly ordered.

Idea$_1$: the solution will also have such a linear order, corresponding to the cyclic ordering instance.

Idea$_2$: only the quartets of the form $ws_i|s_kz$ matter (w ∈ W, z ∈ Z, s_i,s_k ∈ S)

NP-hardness (idea)

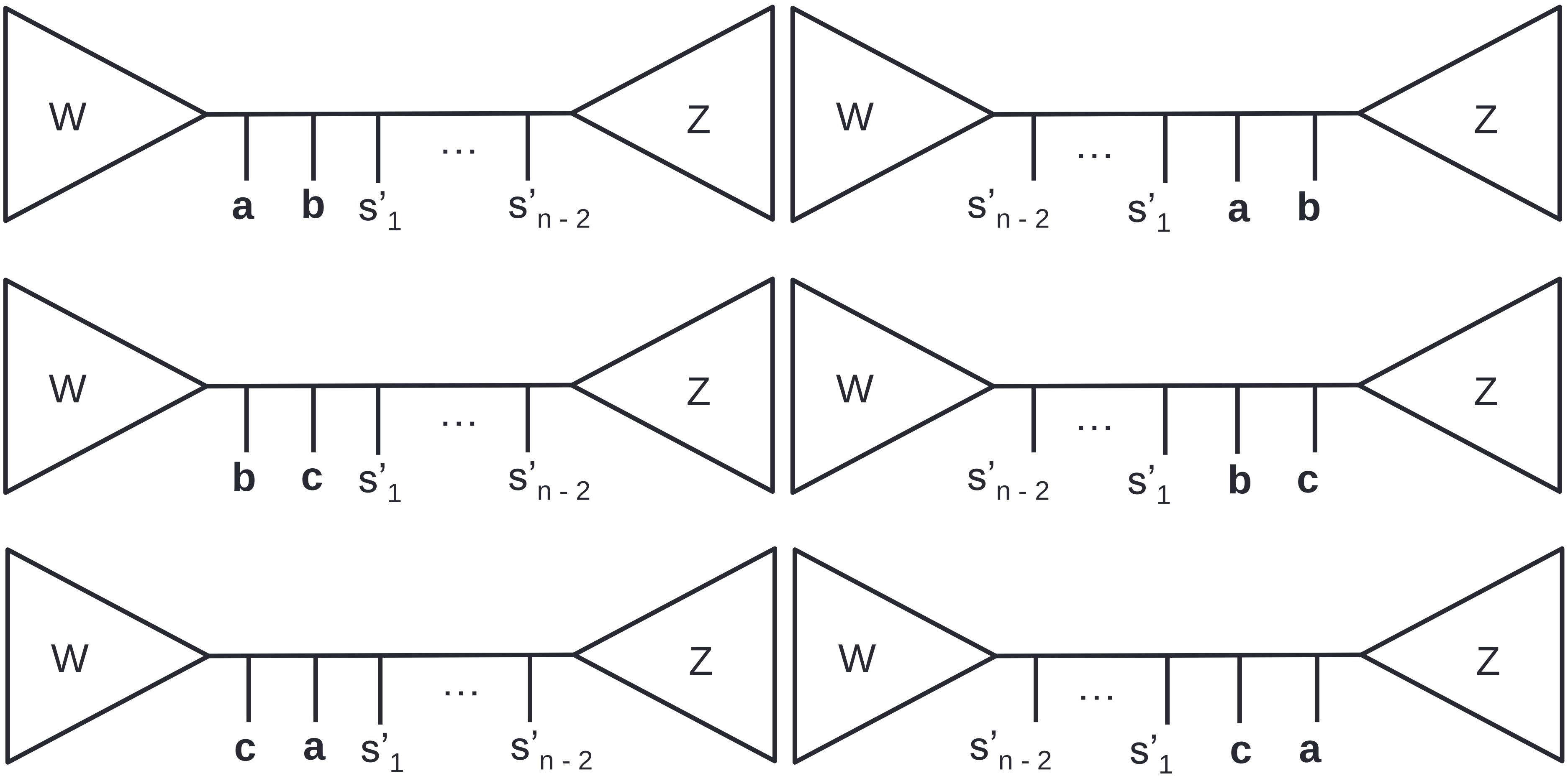
Reduction: main gadget = W-Z trees



Idea$_3$: for each triple (a,b,c) ∈ C, make some W-Z trees that will enforce an optimal tree to order a < b < c or b < c < a or c < b < a in the 'middle'.

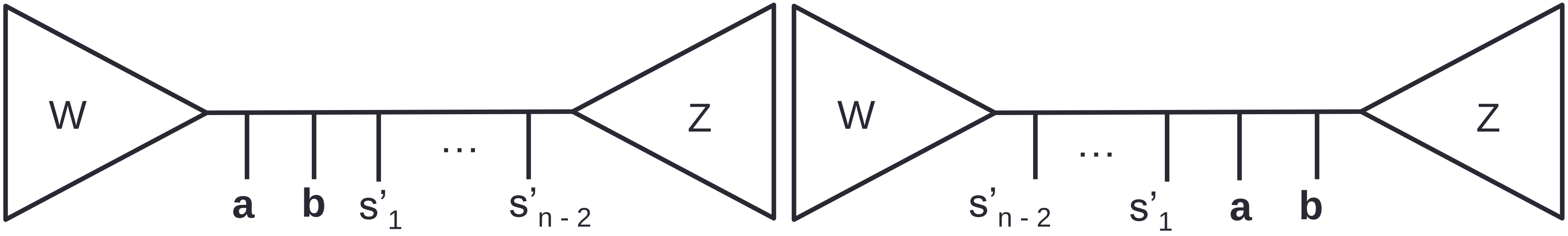
NP-hardness (idea)

(a,b,c) ∈ C => 6 input trees



NP-hardness (idea)

(a,b,c) ∈ C => 6 input trees



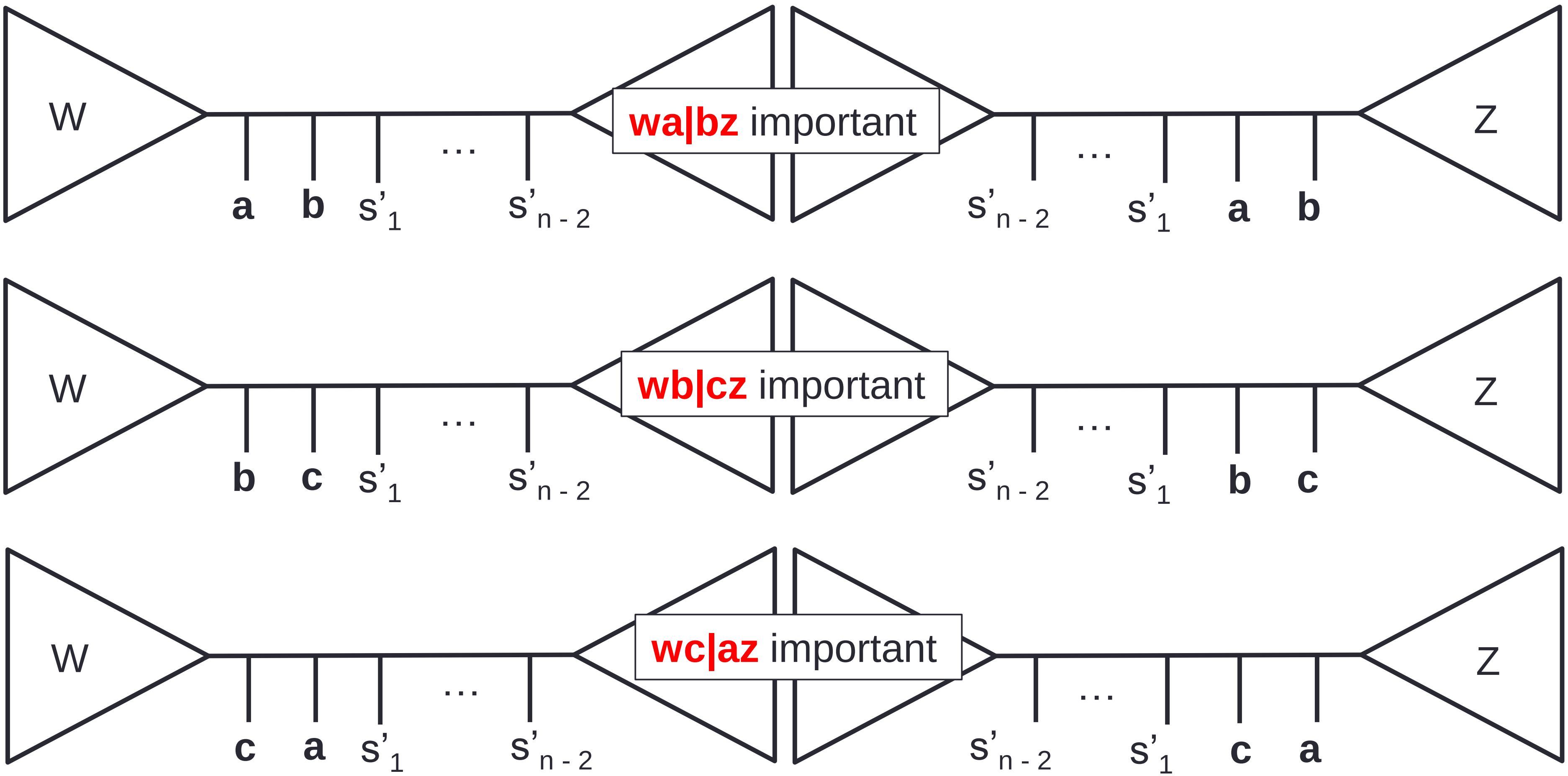
In the first pair of trees, **wa|bz** appears in both trees.

For every other pair s_i, s_k, **both $ws_i|s_kz$ and $ws_k|s_iz$ appear**.

So this tree pair makes **wa|bz, and only wa|bz, important**.

NP-hardness (idea)

(a,b,c) ∈ C => 6 input trees



NP-hardness (idea)

(a,b,c) ∈ C => 6 input trees

wa|bz important

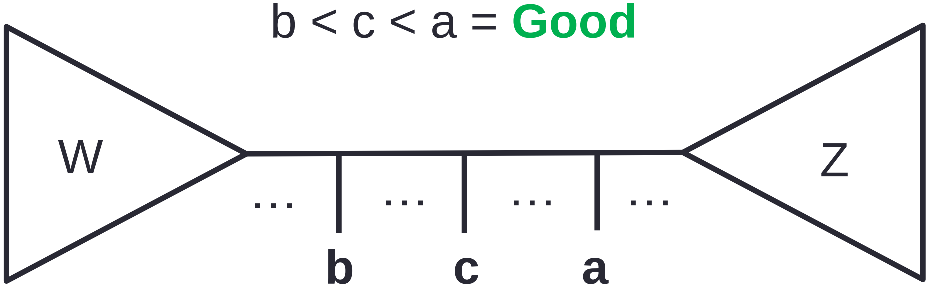
wb|cz important

wc|az important

Ideally, our optimal solution T would contain all important quartets.
This is never possible.
Can we settle for a tree that contains **2 of these important quartets**?

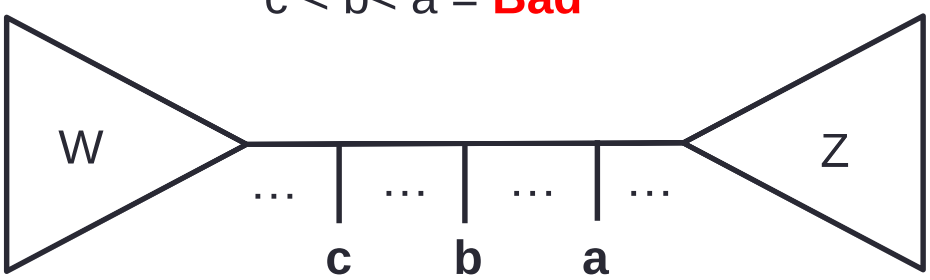
Possible if and only a < b < c, b < c < a or c < a < b

b < c < a = **Good**



Contains **wb|cz** and **wc|az**

c < b< a = **Bad**



Contains only **wc|az**

NP-hardness (idea)

For each triple (a,b,c) ∈ C, make 6 input W-Z trees that will make either
a < b < c or **b < c < a** or **c < b < a** **good** in a solution.

There exists a linear ordering of S satisfying every (a,b,c) ∈ C
if and only if
there exists a tree containing all the **good** orderings
(i.e. containing **2 of the 3 important quartets** defined by each set of 6 trees).

Approximation algorithm

Minimization version of WQC:

Weighted Minimum Quartet Inconsistency (WMQI)

Given: a set of trees T_1, … T_k on the same leafset.
Goal: find a tree T that minimizes the number of quartets of

$$\mathbf{quartets}(T_1) \uplus \ldots \uplus \mathbf{quartets}(T_k)$$

that are not in T (multiple occurrences are counted multiple times).

Approximation algorithm

Minimization version of WQC:

Weighted Minimum Quartet Inconsistency (WMQI)

Given: a set of trees T_1, … T_k on the same leafset.
Goal: find a tree T that minimizes the number of quartets of

quartets(T_1) $\uplus$ … $\uplus$ quartets(T_k)
that are not in T (multiple occurrences are counted multiple times).

Factor 2 approximation:
Return T_i that minimizes the number of quartets not in the input.

Approximation algorithm

Min-version WMQI has a 2-approximation (rejects, at worst, twice too many quartets)

Max-Version WQC has a Randomized 1/3-approximation: generate a random tree.

- Each input quartet has a 1/3 chance of being in the tree.
- More than just a 1/3-approx. : the random tree contains 1/3 of the input quartets (on expectation).

Lemma: for WQC, this is a 1/2-approximation algorithm:

Return the best of the WMQI 2-approx. or the WQC 1/3-approx.

Proof idea: if the WQC solution preserves 2/3 or less of the input quartets, the tree that contains 1/3 of the input quartets is a 1/2-approx.
If the WQC solution preserves more than 2/3 of the input quartets, the WMQI 2-approx. rejects few quartets, and so preserves many: at least 1/2 as many as the optimal solution (details => paper).

Approximation algorithm

Annoying problem: the 1/3-approximation is a randomized algorithm – it only preserves 1/3 of the input quartets **on expectation**.

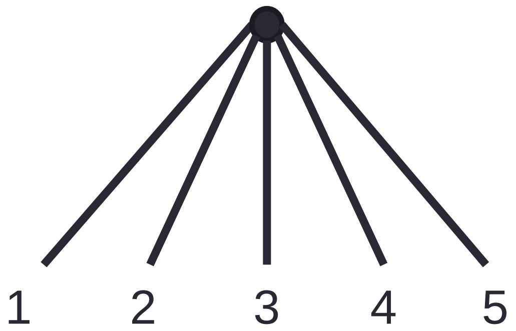
We derandomize this algorithm, using the method of conditional expectation.

Derandomization

Assume the output tree is rooted (not a technical problem).

Start with a completely unresolved tree.
If we resolve (i.e. binarize) randomly, each quartet appears with 1/3 probability.

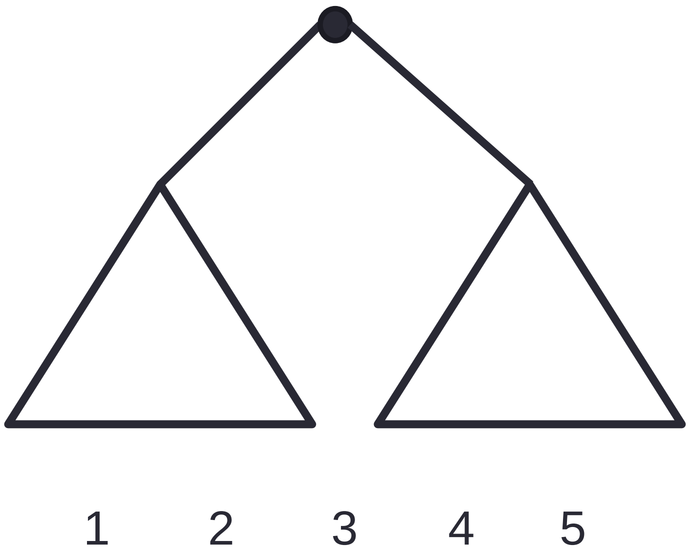
12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45



Derandomization

Let us consider the first split of a solution.
Idea: we will look for a split that preserves 1/3 quartets, on expectation (we know such a split exists).

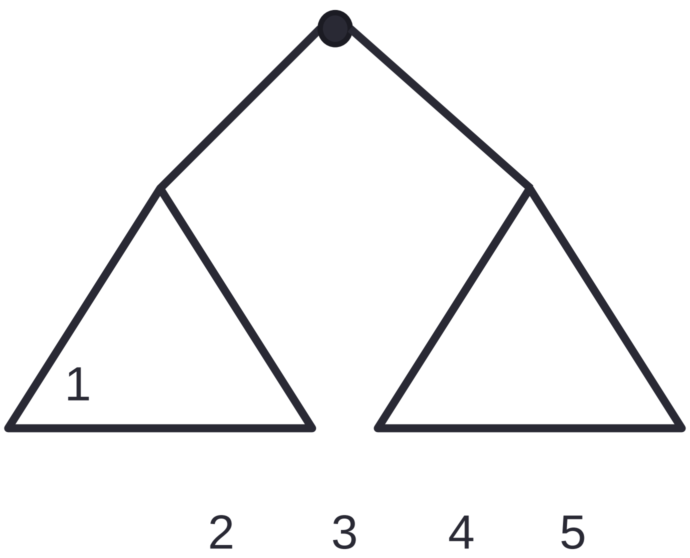
12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45



Derandomization

Let us consider the first split of a solution.
Idea: we will look for a split that preserves 1/3 quartets, on expectation (we know such a split exists).
Let's put 1 on the left (this choice is arbitrary).

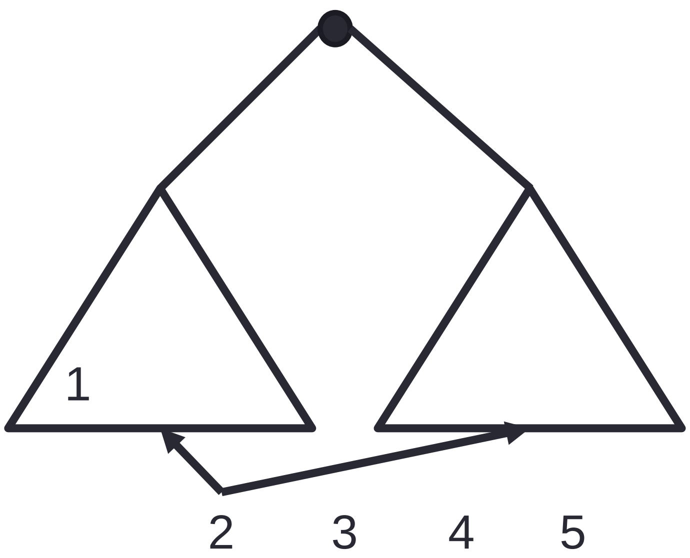
12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45



Derandomization

Let us consider the first split of a solution.
Idea: we will look for a split that preserves 1/3 quartets, on expectation (we know such a split exists).
Let’s put 1 on the left (this choice is arbitrary).
Now where should 2 go? **Try both!**

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

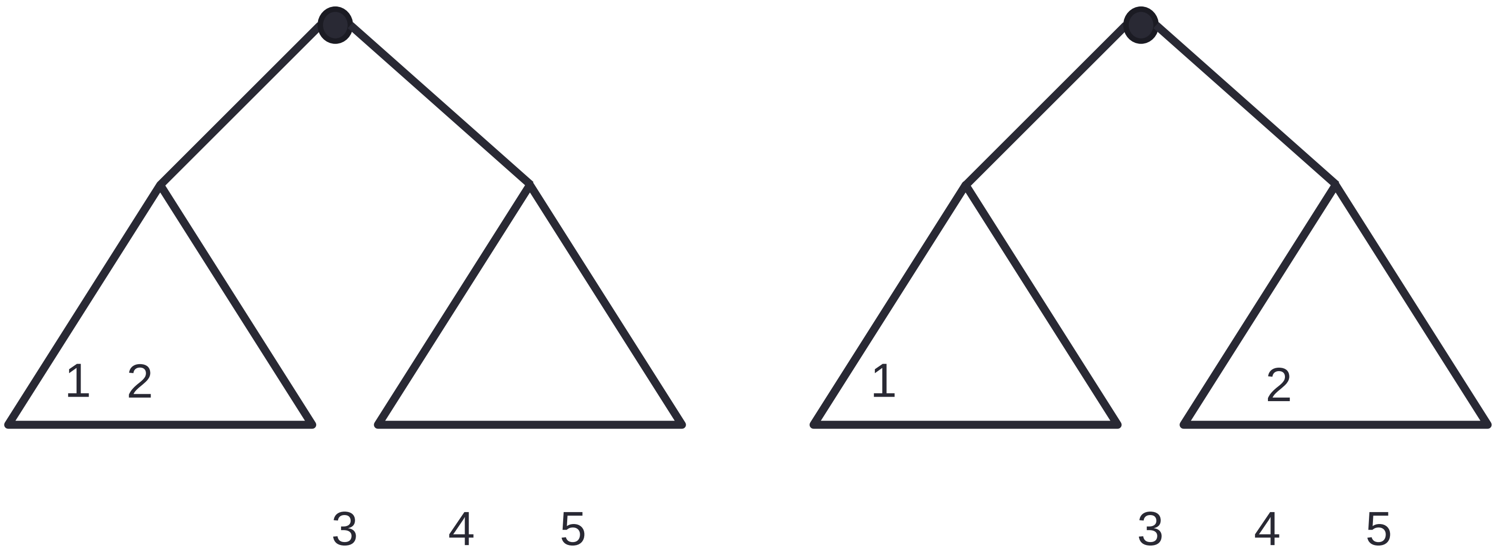


Derandomization

For each of the 2 options, view {3,4,5} as randomly placed into either split with probability ½.

Then we randomly resolve each "random split", yielding a probability on every input quartet.

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

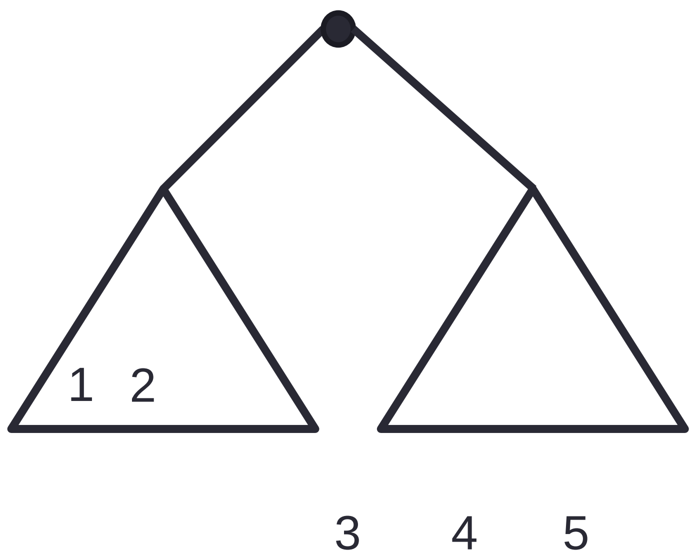


Derandomization

For example, in this scenario,

Pr[T contains 12|34] = Pr[3,4 go left AND left split resolves into 12|34] +
Pr[3 goes left, 4 goes right AND left split resolves into 12|3] +
Pr[3 goes right, 4 goes left AND left split resolves into 12|4] +
Pr[3,4 go right]

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

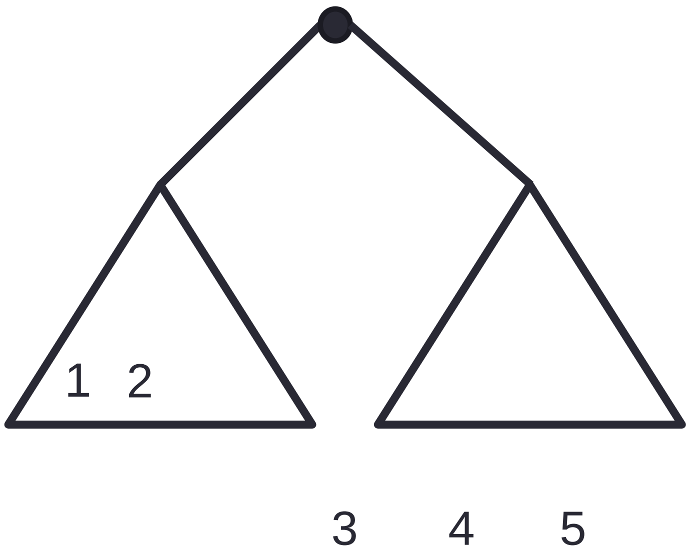


Derandomization

For example, in this scenario,

Pr[T contains 12|34] = Pr[3,4 go left AND left split resolves into 12|34] +
Pr[3 goes left, 4 goes right AND left split resolves into 12|3] +
Pr[3 goes right, 4 goes left AND left split resolves into 12|4] +
Pr[3,4 go right]
= 1/4 * 1/3 + 1/4 * 1/3 + 1/4 * 1/3 + 1/4 = 1/2

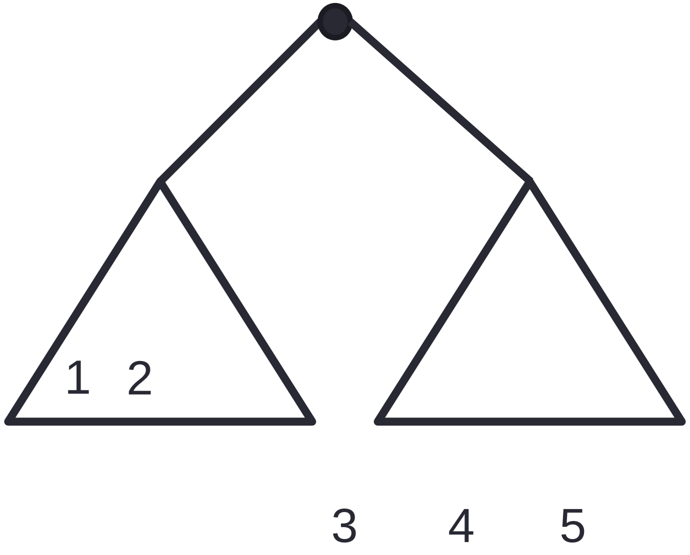
12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

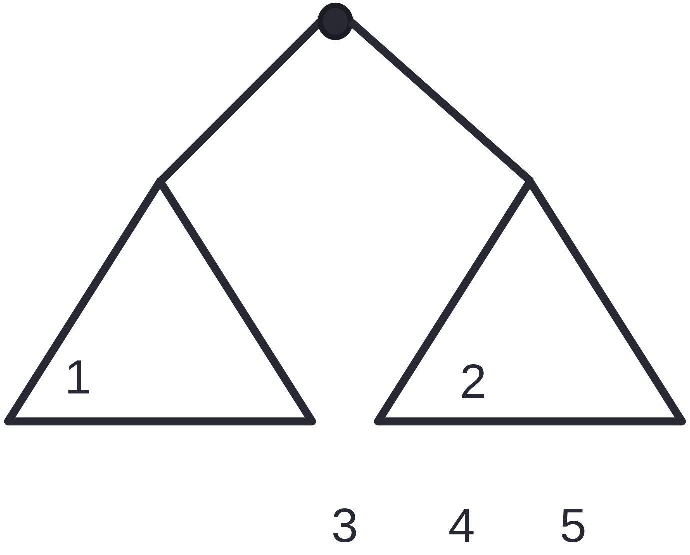


Derandomization

For each scenario, compute $\sum_{q \in Q} Pr[T \textit{ contains } q]$, and keep the scenario with maximum value.
One of these two “partial splits” will contain at least 1/3 of the input quartets, on expectation.

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

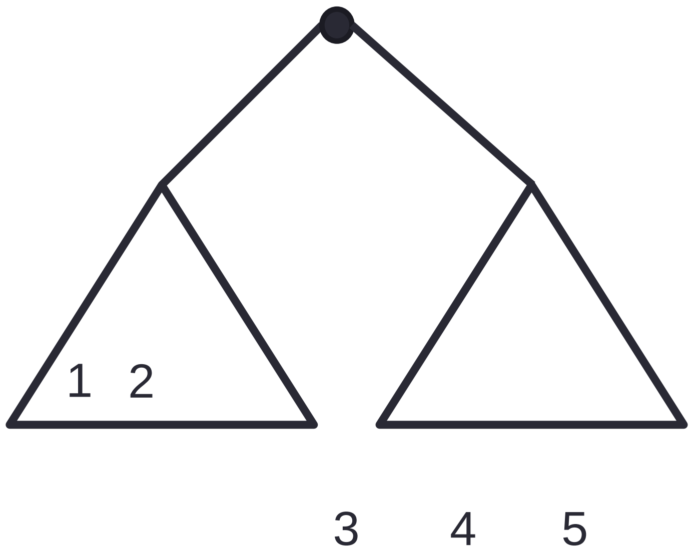




Derandomization

For each scenario, compute $\sum_{q \in Q} \boldsymbol{Pr[T\ contains\ q]}$, and keep the scenario with maximum value.
One of these two “partial splits” will contain at least 1/3 of the input quartets, on expectation.
Once a choice is made, proceed with the next label.

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45



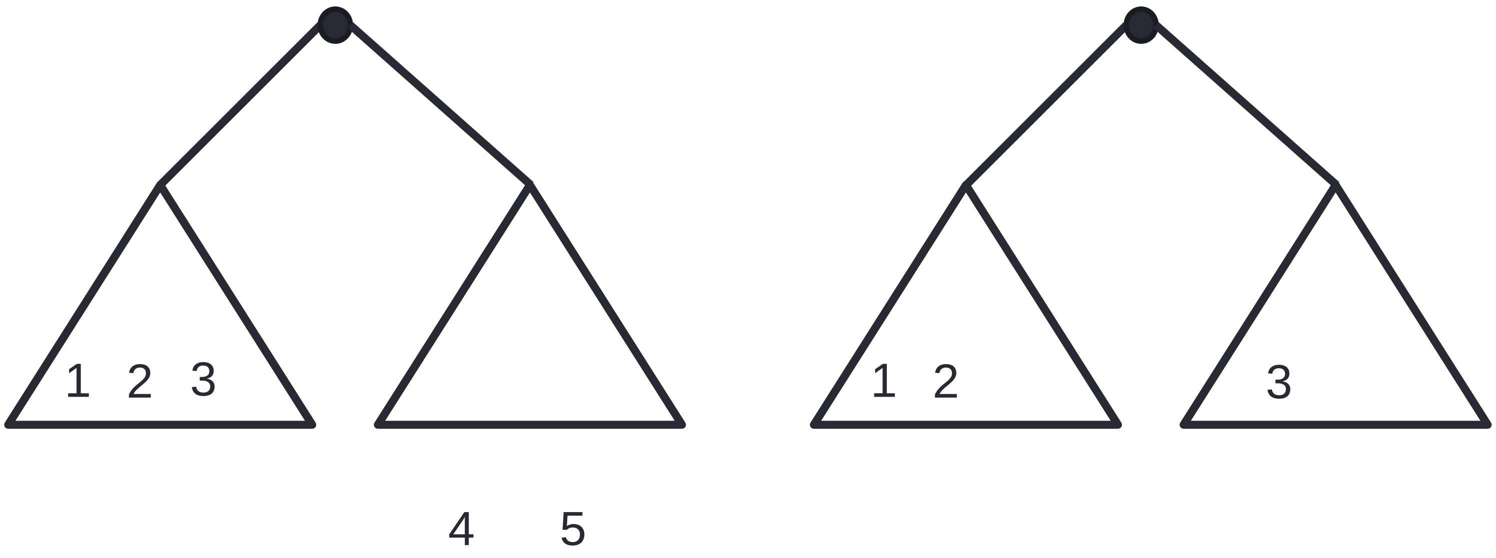
Derandomization

For each scenario, compute $\sum_{q \in Q} Pr[T \; contains \; q]$, and keep the scenario with maximum value.

One of these two “partial splits” will contain at least 1/3 of the input quartets, on expectation.

Once a choice is made, proceed with the next label.

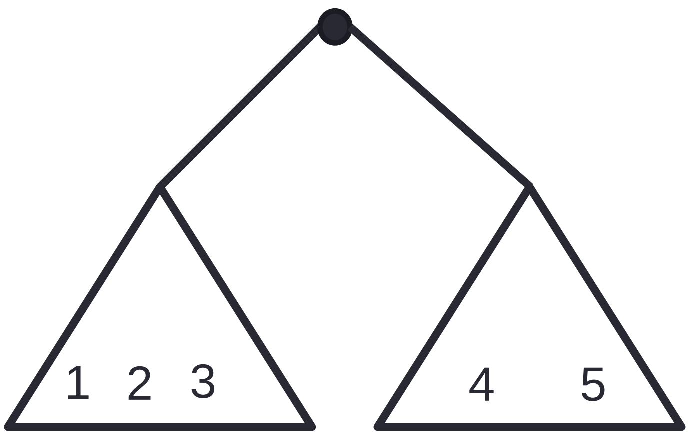
12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45

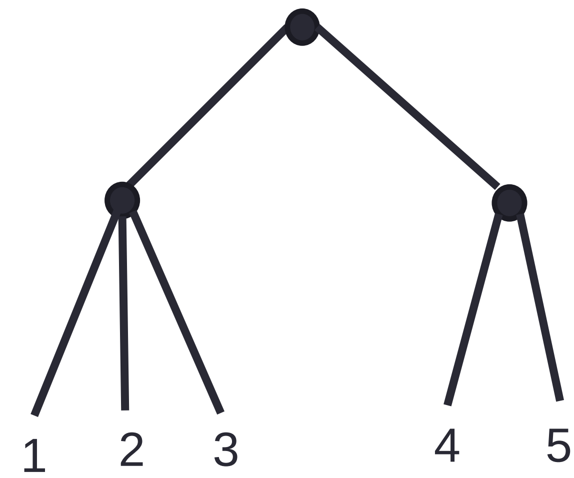


Derandomization

Once a split is complete, repeat recursively on the two splits.

12|34
12|35
12|45
23|45
13|42
13|45
13|25
34|25
13|24
13|25
13|45
32|45





Derandomization

At each decision we make in this algorithm, we maintain a conditional expectation of at least 1/3
$\Rightarrow$ When we finish with a binary tree, it contains 1/3 quartets from the input.

Takes time $O(k^2n^2 + kn^4 + n^5)$

(n is the number of leaves, k is the number of input trees)

Conclusion

Are there better approximation algorithms (with a ratio above ½) ?

Is the WQC problem Max SNP-Hard?

What is the approximation ratio of the ASTRAL algorithm? (see paper)

Fixed parameter tractability of WQC?
Any way to obtain an exact solution in reasonable time?

- Not too hard that the problem is FPT w.r.t. parameter
  q = # of quartets to discard from the input multiset
- Take time $O^*(4^q)$, which is not reasonable.