A Randomized Approximation Algorithm for rSPR Distance

Zhi-Zhong Chen

zzchen@mail.dendai.ac.jp

Tokyo Denki University

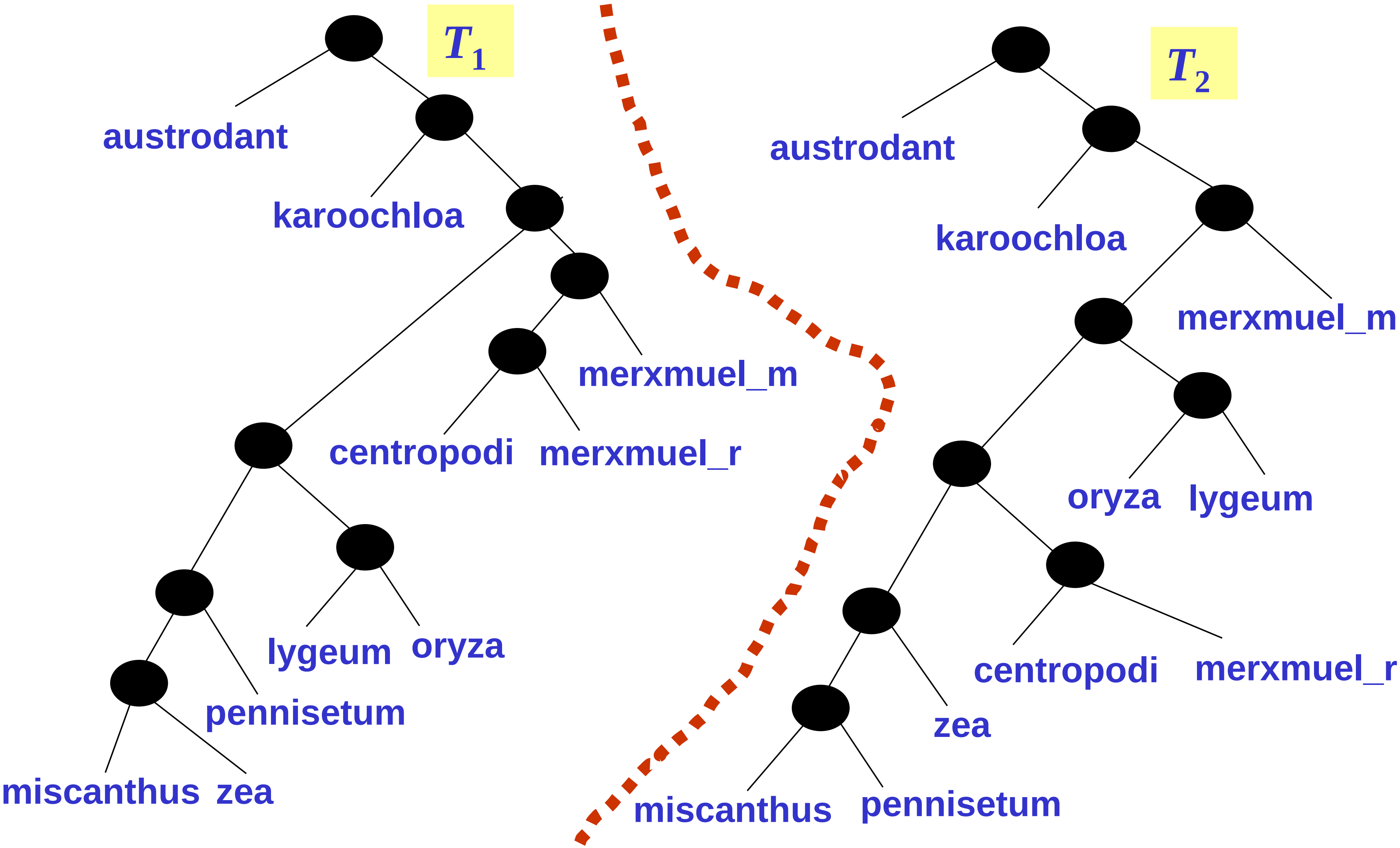
A new version for an expected ratio of 27/11 < 2.5

The rSPR distance between two binary trees

- Given two binary phylogenetic trees, we want to remove as few edges as possible from the trees so that they become the same forest.

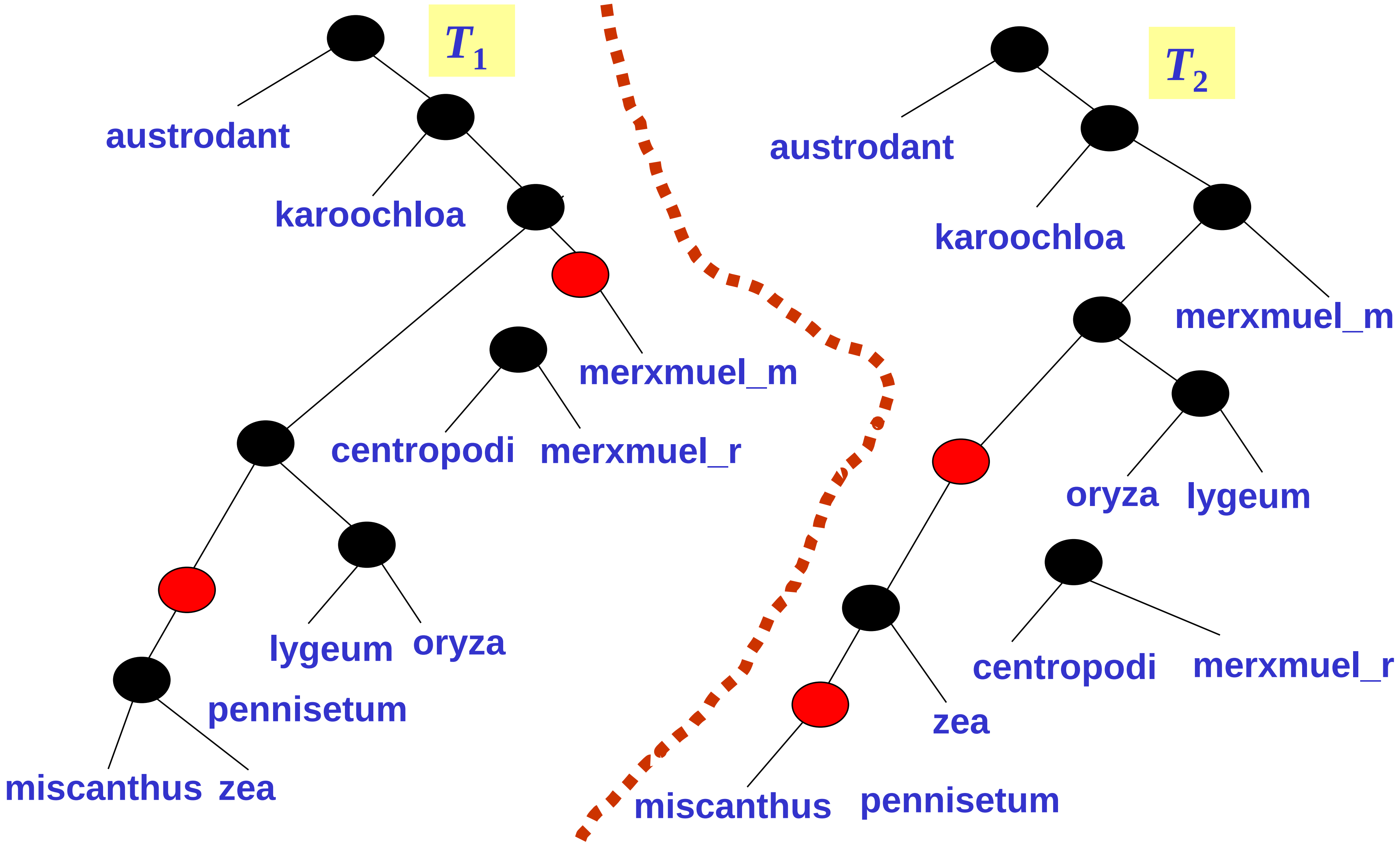
  The minimum number of removed edges is the rSPR distance between the two trees.

Example



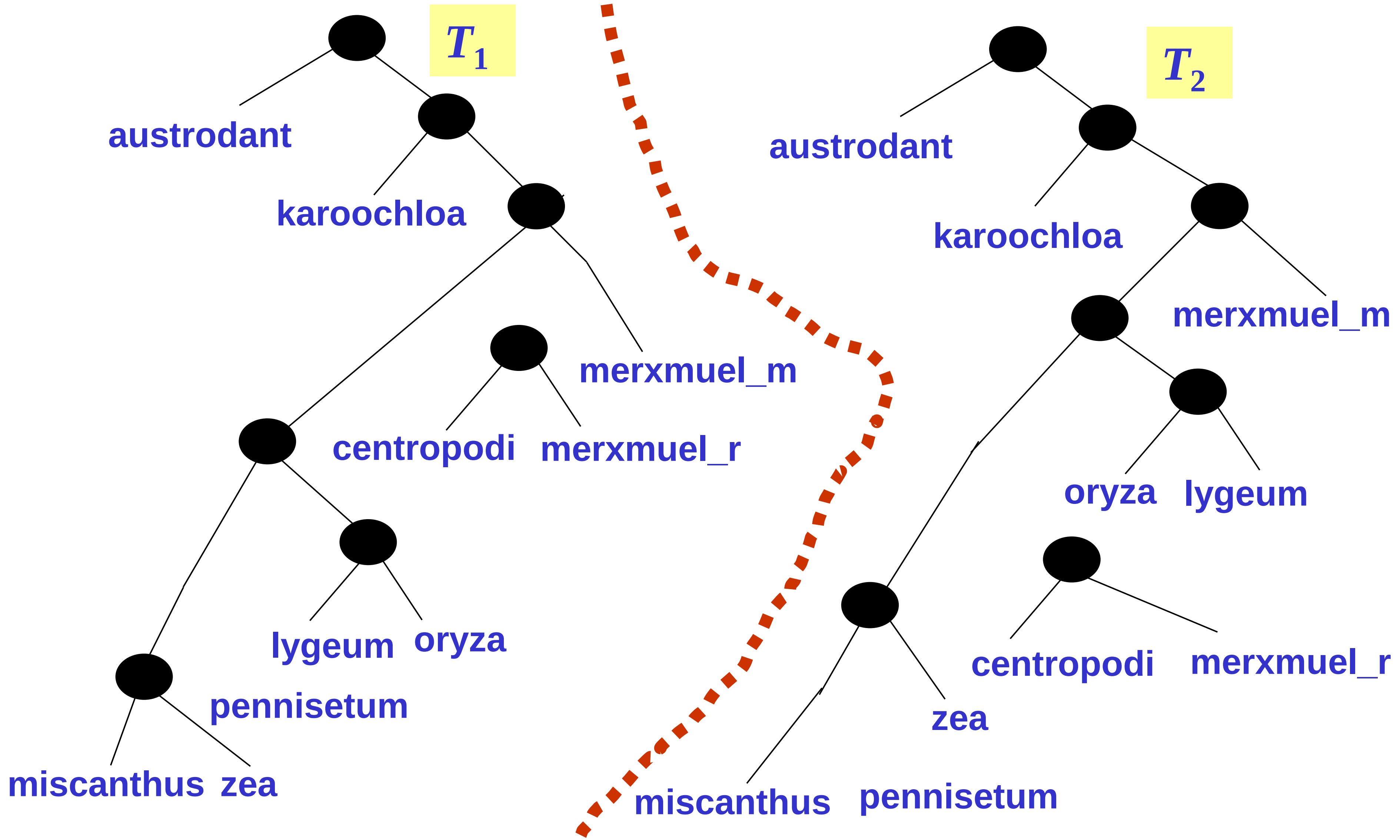
We want to delete as few edges as possible from the trees so that they become the same forest.

Example (continued)



If we contract all the degree-1 vertices, then the forests become the same.

Example (continued)



If we contract all the degree-1 vertices, then the forests become the same.

Previous work

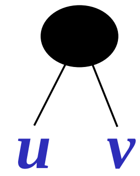
- **Hein, et al., 1996: The problem is NP-hard.**
- **Hein, et al., 1996: The first approximation Algorithm.**
  Rodrigues, et al., 2007: Ratio 3 and quadratic time.
  Whidden, et al., 2009: Ratio 3 and linear time.
  Shi, et al., 2009: Ratio 2.5 and quadratic time.

  Not aware of before the workshop!!!

- **Fixed-parameter algorithms:**
  Take the rSPR distance *d* as the parameter and strive to design an algorithm whose complexity is exponential only in *d*.
- **Approximation algorithms for rSPR distance have been used to speed up fixed-parameter algorithms for rSPR distance and are also used to speed up fixed-parameter algorithms for hybridization number and reticulate networks.**

The simplest exact algorithm

① Find two sibling leaves in u and v in T_2.



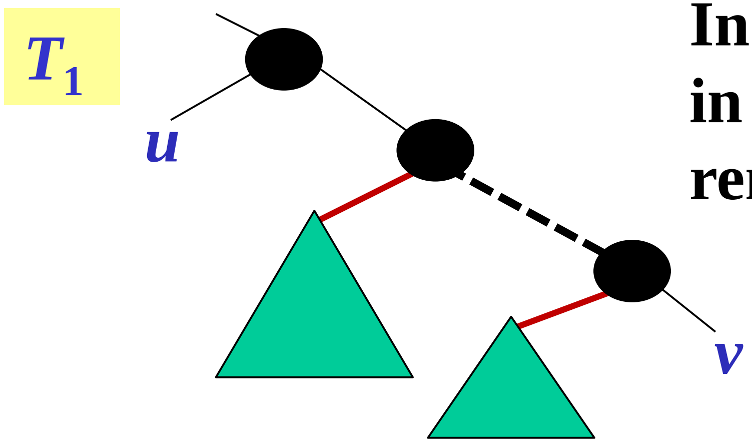
② If u and v are also siblings in T_1, then merge them into a single leaf in both trees and repeat.

③ If u and v are not siblings in T_1, then there are two cases:

Case 1: u and v are in different connected components in T_1.

In this case, we have two choices: isolate u or v.

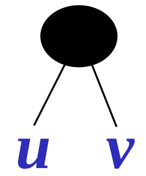
Case 2: u and v are in the same connected component in T_1.



In this case, other than the two choices in Case 1, we have another choice of removing the brown edges.

The ratio-3 approximation algorithm

① Find two sibling leaves in u and v in T_2.



② If u and v are also siblings in T_1, then merge them into a single leaf in both trees and repeat.

③ If u and v are not siblings in T_1, then there are two cases:

Case 1: u and v are in different connected components in T_1.

In this case, we isolate both u and v.

Each of the two removed edges receives a penalty of 1 / 2.

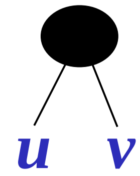
Invariant 1: (decrement of rSPR distance) ≥ (new total penalty)

Invariant 2: Never penalize an edge twice or more.

Approximation ratio: 1 / (the smallest penalty received by an edge)

The ratio-3 approximation algorithm

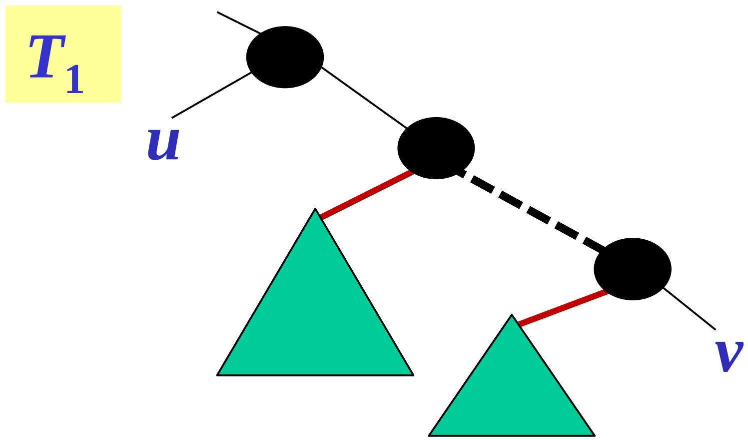
① Find two sibling leaves in u and v in T_2.



② If u and v are also siblings in T_1, then merge them into a single leaf in both trees and repeat.

③ If u and v are not siblings in T_1, then there are three cases:

Case 2: u and v are in the same connected component in T_1.



In this case, other than isolating both u and v as in Case 1, we also remove an arbitrary brown edge.

Each of the three removed edges receives a penalty of $1/3$.

Invariant 1: (decrement of rSPR distance) $\geq$ (new total penalty)

Invariant 2: Never penalize an edge twice or more.

The ultimate question

The ratio 3 has remained the best for about a decade.

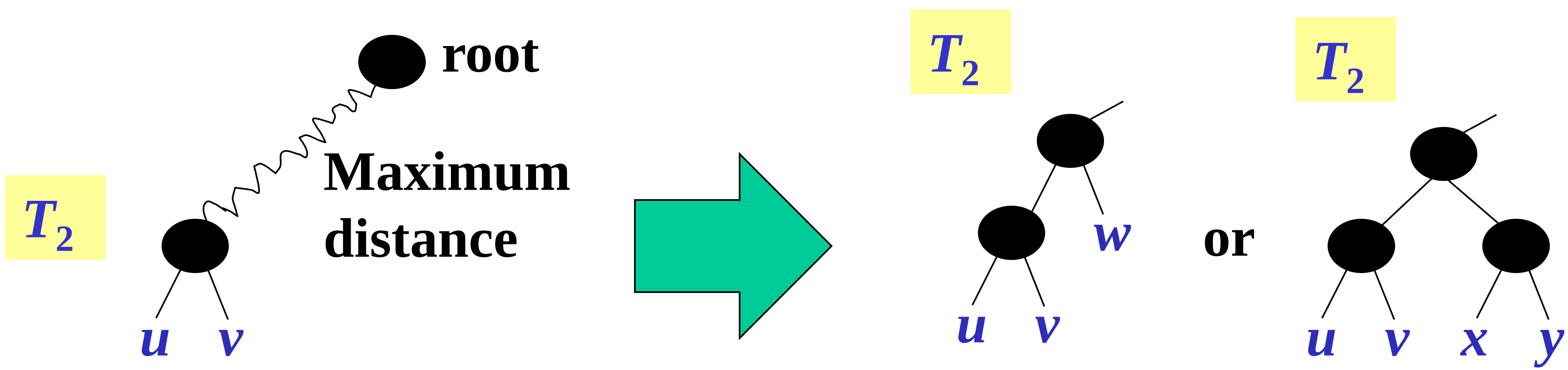
Can we achieve a better ratio than 3?

Shi et al. , 2014: Yes!

Ideas behind Shi et al.'s algorithm

Basic idea: Rather than looking at a single sibling-leaf pair in T_2, look wider.

Idea 1: Start at a sibling leaf pair in T_2 whose distance from the root is maximized.



Depending on how u, v, and w or x, y are related in T_1, there are a lot of cases.

Idea 2: Show that all the cases lead to a ratio of at most 2.5.

Note: The ideas of looking wider and finding a sibling-leaf pair as above were previously used in our exact algorithm [Chen et al., 2013].

Shi et al.'s open question

Shi et al., 2014 : Can we achieve a better ratio than 2.5?

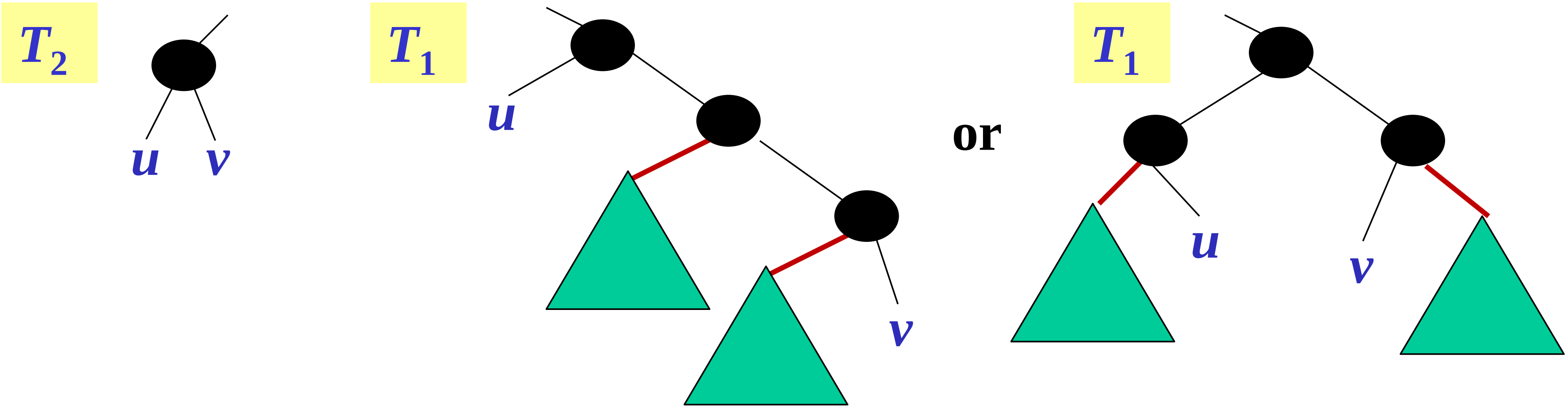
Our answer: Yes!

but with the help of randomness!

The expected ratio is 27/11.

How to improve Shi et al.'s algorithm?

A simple illustrative Case: The distance between u and v in T_1 is 4.



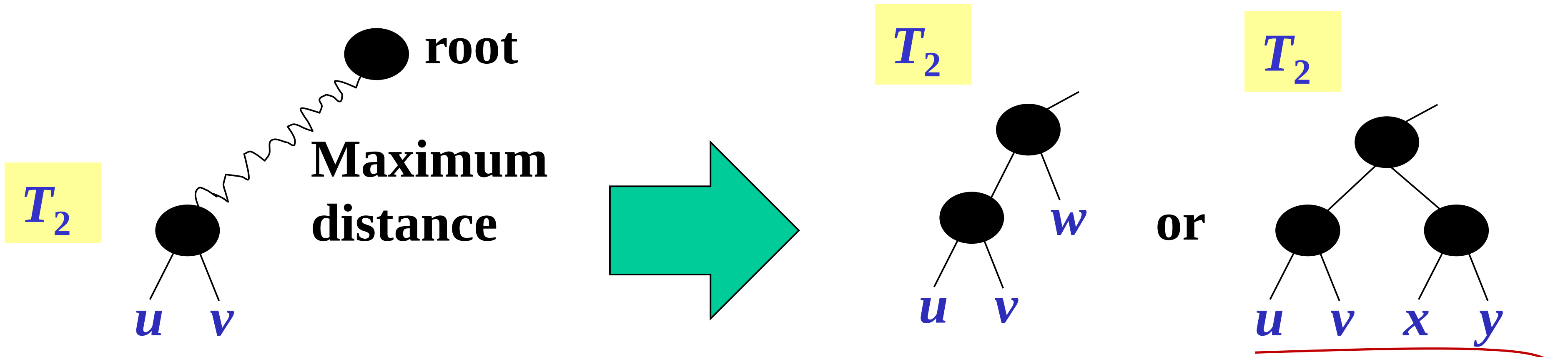
In this case, we delete the two **brown edges** in T_1,
and then merge u and v into a single leaf in both T_1 and T_2.

Each of the two removed edges receives a penalty of 1 / 2.

Invariant 1: (decrement of rSPR distance) $\geq$ (new total penalty)

Invariant 2: Never penalize an edge twice or more.

How to improve Shi et al.'s algorithm? -- continued



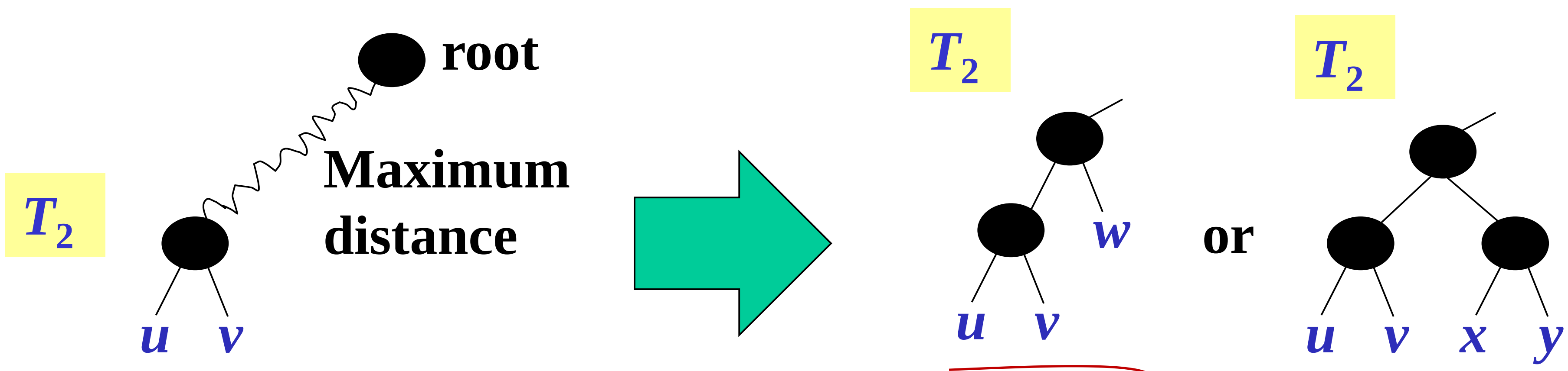
Idea 1: **For this pattern, we can show (by a careful case analysis) that in those cases where Shi et al. only get a ratio of 2.5, we can actually delete at most 7 edges from T_1 so that the decrement of rSPR distance between T_1 and T_2 is at least 3.**

Each of the removed edges receives a penalty of 3 / 7.

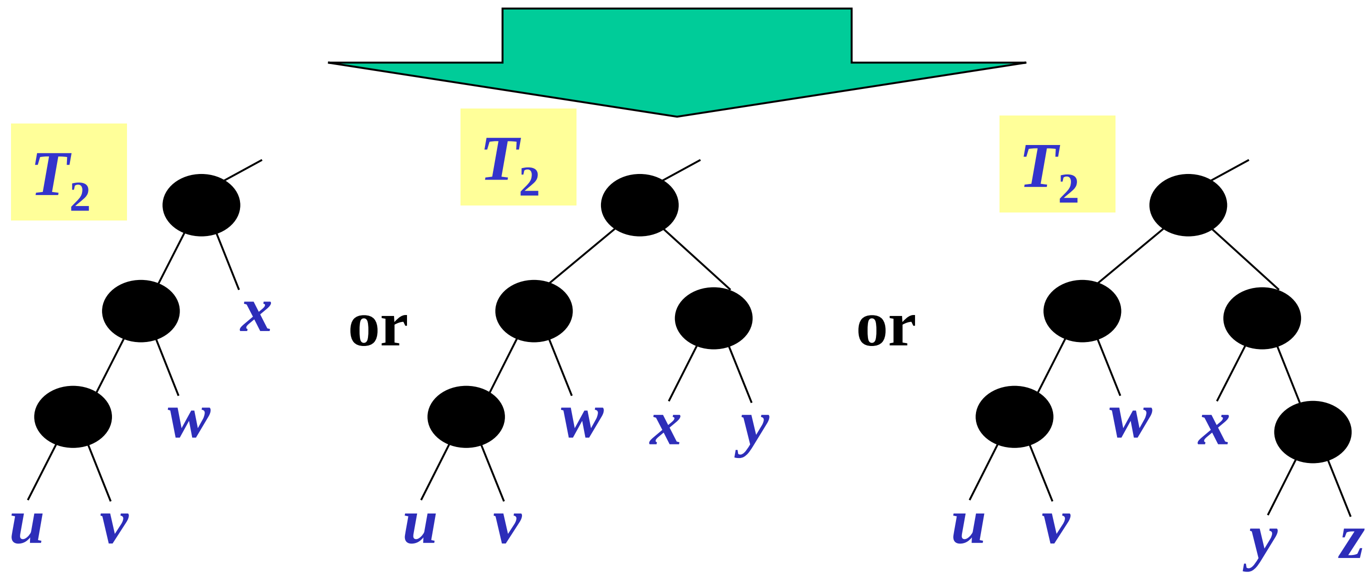
Invariant 1: (decrement of rSPR distance) ≥ (new total penalty)

Invariant 2: Never penalize an edge twice or more.

How to improve Shi et al.'s algorithm? -- continued

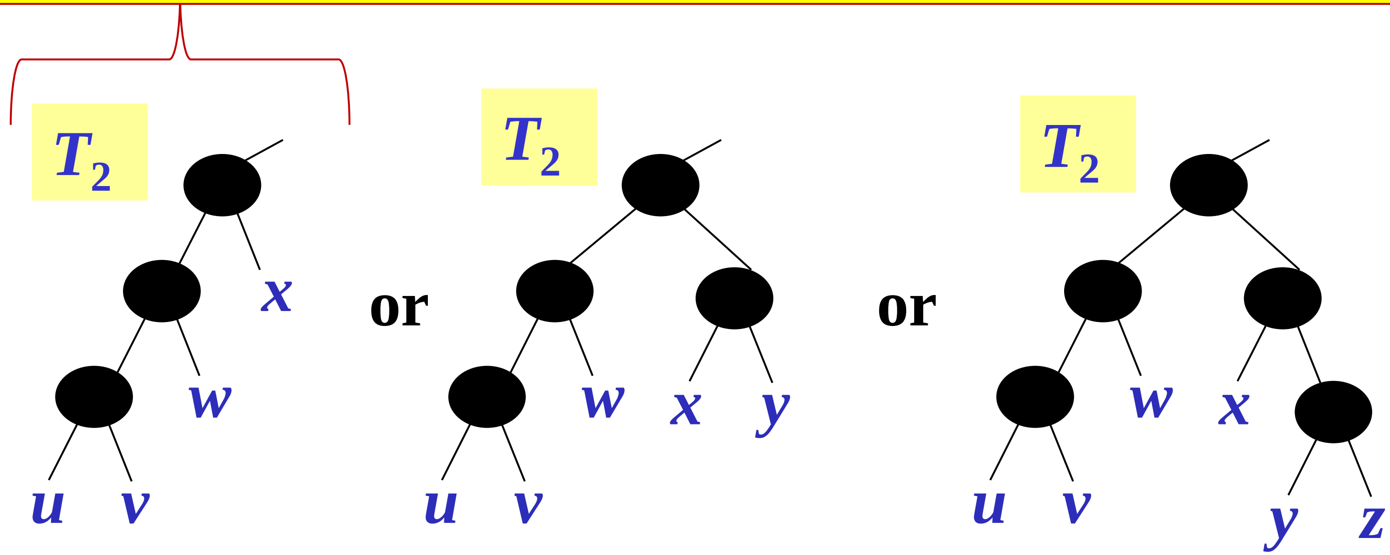


Idea 2: **For this pattern, in order to obtain a ratio better than Shi et al.'s 2.5, we have to look even wider!!!**



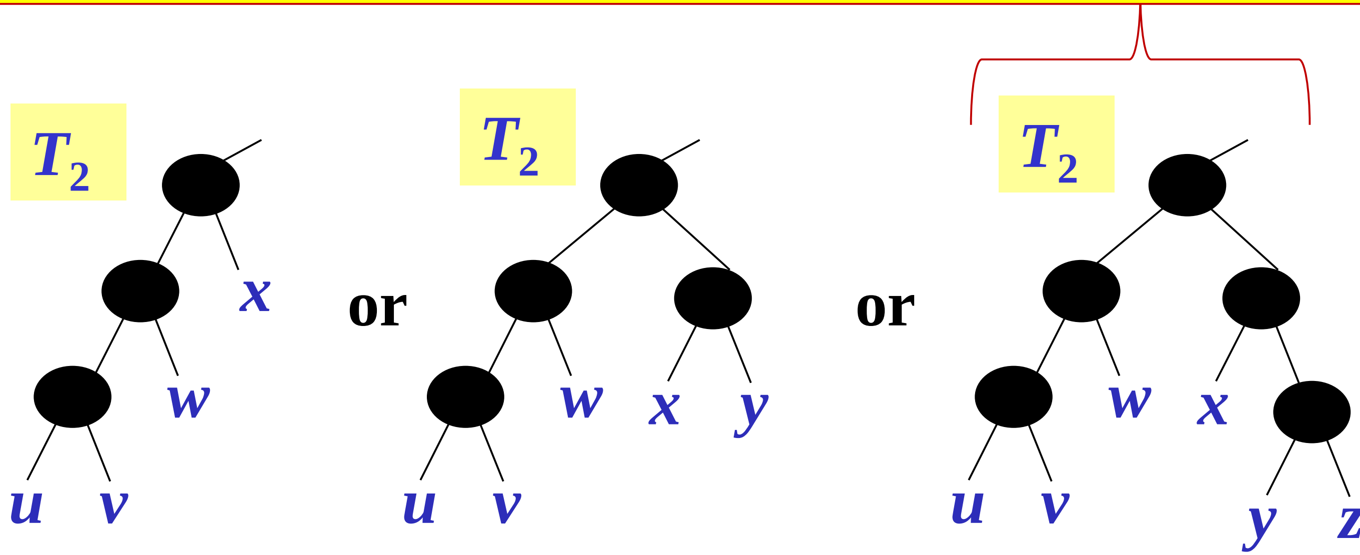
How to improve Shi et al.'s algorithm? -- continued

Idea 3: For this pattern, we can show (by a careful case analysis) that in the worst case, we can delete at most **7** edges from T_1 so that the decrement of rSPR distance between T_1 and T_2 is at least **3**.



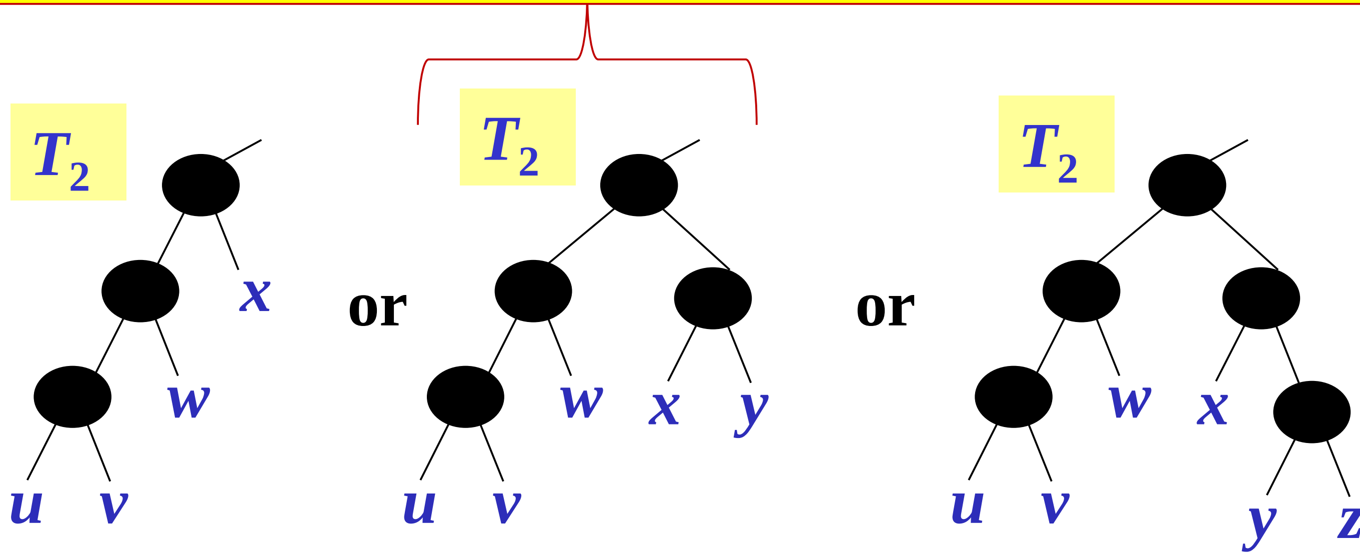
How to improve Shi et al.'s algorithm? -- continued

Idea 4: For this pattern, we can show (by a careful case analysis) that in the worst case, we can delete at most **12** edges from T_1 so that the decrement of rSPR distance between T_1 and T_2 is at least **5**.

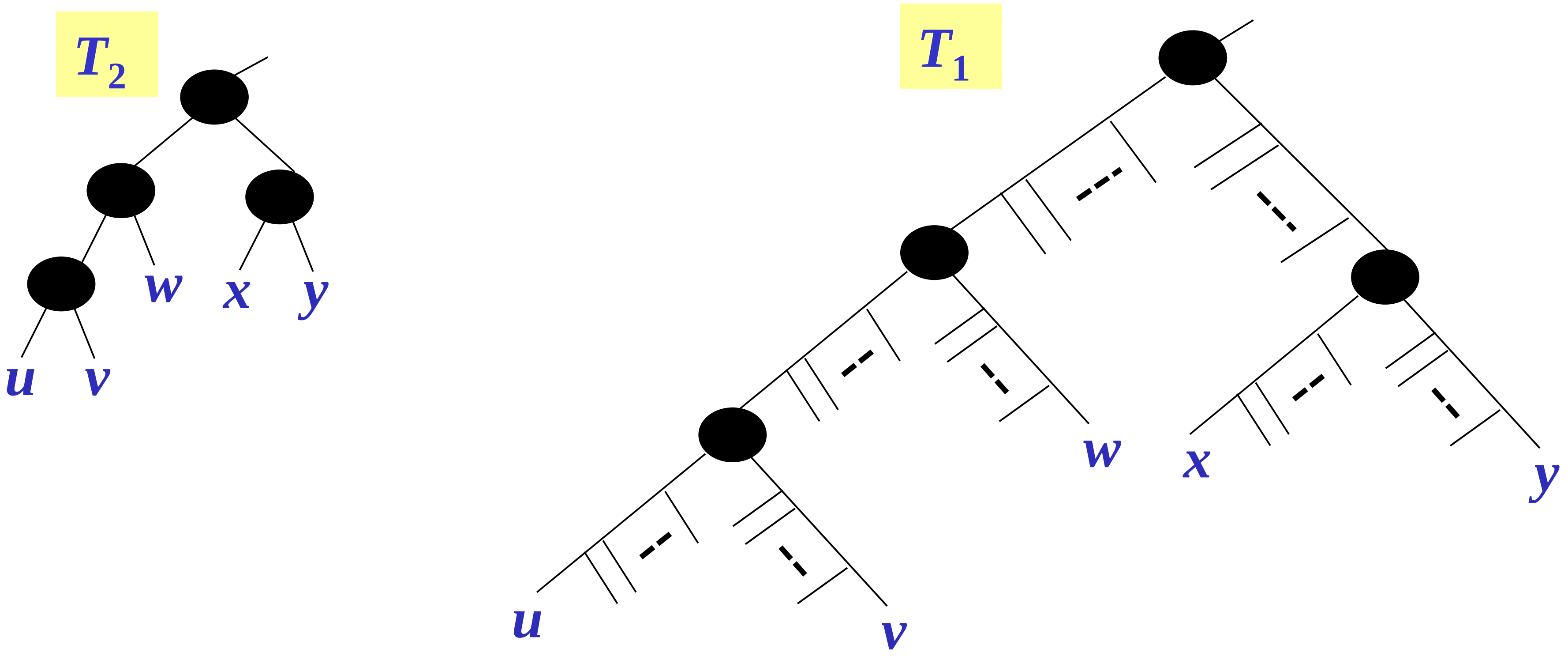


How to improve Shi et al.'s algorithm? -- continued

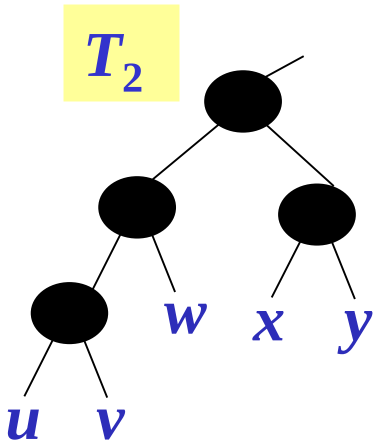
Unfortunately, **for this pattern, there are cases (called bad cases) where we cannot get a ratio better than 2.5.**

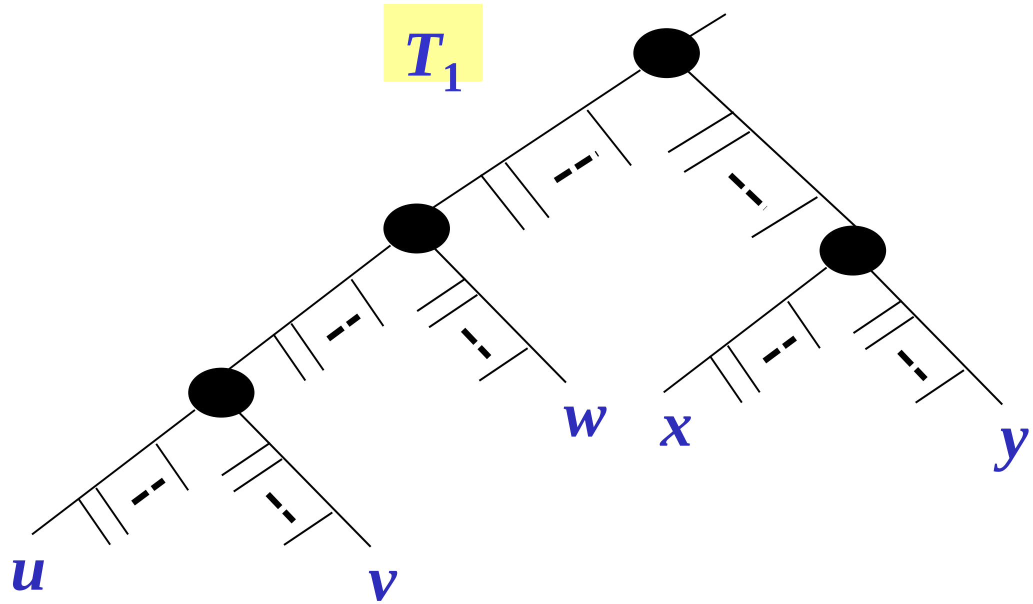


An example bad case



How to deal with the bad cases? -- example





We show that we can find three sets C_1, C_2, and C_3 of edges in T_1 s.t.

(1) the size of each C_i is 9,

(2) deleting the edges in each C_i from T_1 decreases the rSPR distance between T_1 and T_2 by at least 3, and

(3) if we select one C_i among C_1, C_2, and C_3 uniformly at random and delete the edges in C_i from T_1, then the rSPR distance between T_1 and T_2 decreases by at least 4 with probability at least 2/3.

How to deal with the bad cases (size 5)? --Continued

The expected ratio is: $$\frac{9}{4 \times \frac{2}{3} + 3 \times \frac{1}{3}} = \frac{27}{11} < 2.5$$

We show that we can find three sets $C1$, $C2$, and $C3$ of edges in T_1 s.t.

(1) the size of each Ci is 9,

(2) deleting the edges in each Ci from T_1 decreases the rSPR distance between T_1 and T_2 by at least 3, and

(3) if we select one Ci among $C1$, $C2$, and $C3$ uniformly at random and delete the edges in Ci from T_1, then the rSPR distance between T_1 and T_2 decreases by at least 4 with probability at least 2/3.

Open problems

1. **Better algorithms?**
2. **Multiple trees?**
3. **Non-binary trees?**