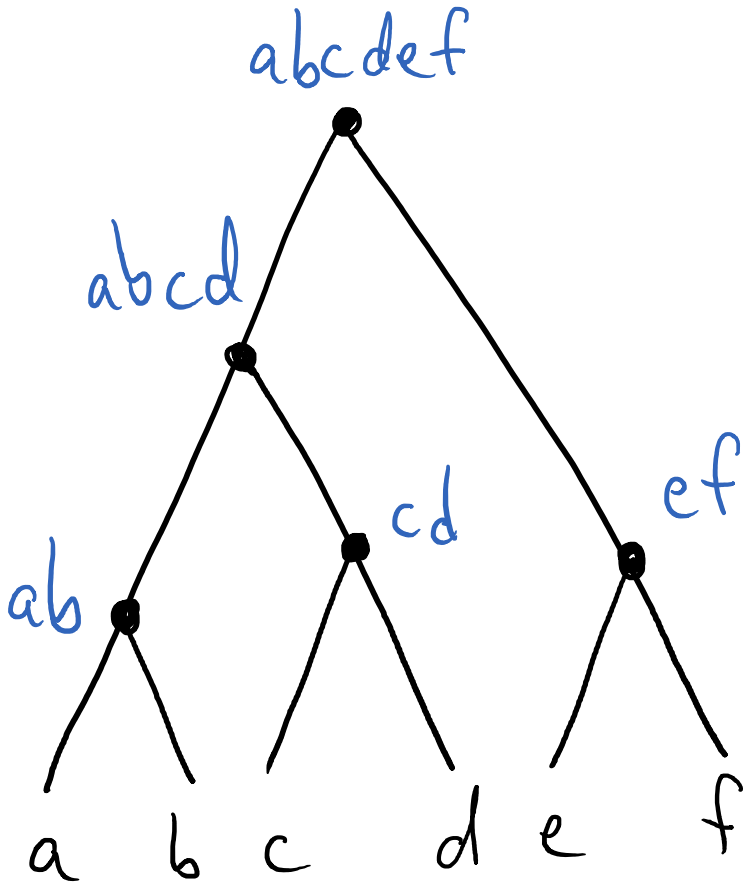
Constructing incompatibility graphs of trees in optimal output-sensitive time

Manuel Lafond **Université de Sherbrooke**

Clusters in trees

Given a (rooted) tree T and a node $v \in T$, $cluster_T(v)$ is the set of leaf taxa that descend from v.

$clusters(T) = \{cluster_T(v) : v \in V(T)\}$



Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

$A = \{a, b, c, d, e\}$

$B = \{b, d, e\}$

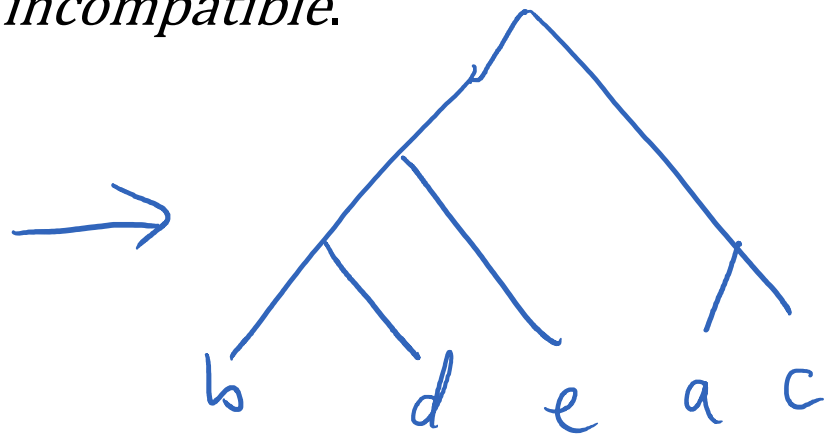
Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

$A = \{a, b, c, d, e\}$

$B = \{b, d, e\}$



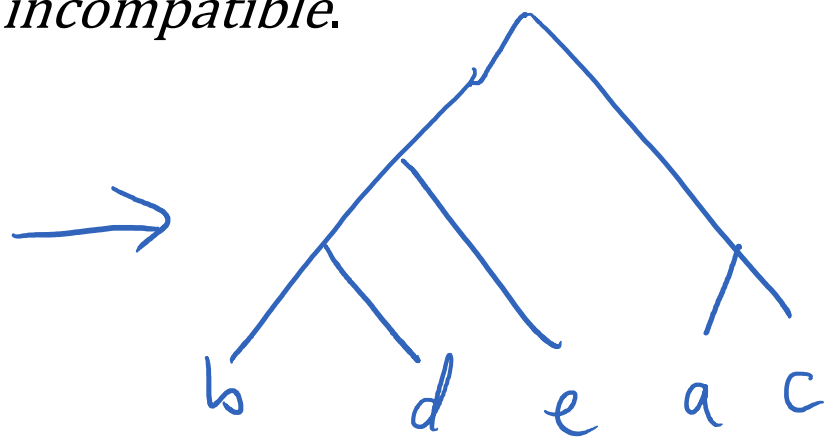
Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

$A = \{a, b, c, d, e\}$

$B = \{b, d, e\}$



$A = \{a, b, c\}$

$B = \{b, c, d\}$

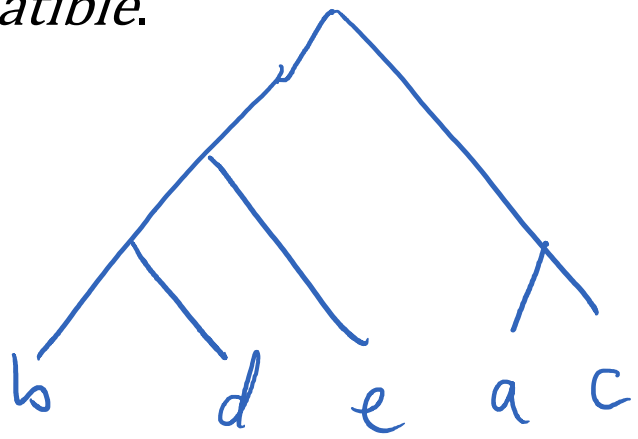
Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

$A = \{a, b, c, d, e\}$

$B = \{b, d, e\}$ $\longrightarrow$



$A = \{a, b, c\}$

$B = \{b, c, d\}$ $\longrightarrow$

Incompatible clusters

Let A, B be two sets of taxa. We say that A, B are *compatible* if there exists a tree that contains both A and B as clusters.

Otherwise, A, B are *incompatible*.

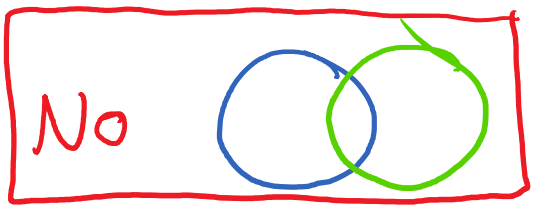
Classic problem: given a collection of sets of taxa $A = \{A_1, \ldots, A_k\}$, reconstruct a tree whose clusters are exactly A, if possible.

Theorem: feasible if and only if sets in A are pairwise-compatible.

Theorem: A, B are compatible if and only if one of the following holds: either

1. $A \subseteq B$;
2. $B \subseteq A$; or
3. $A \cap B = \emptyset$.

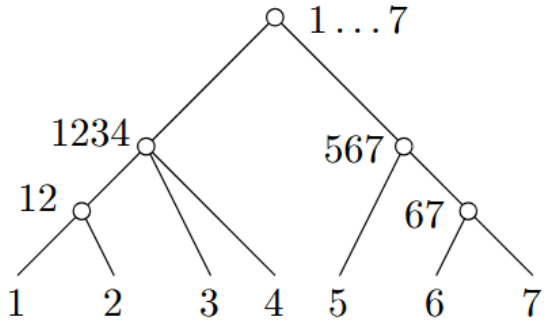
(i.e., there is no overlap)

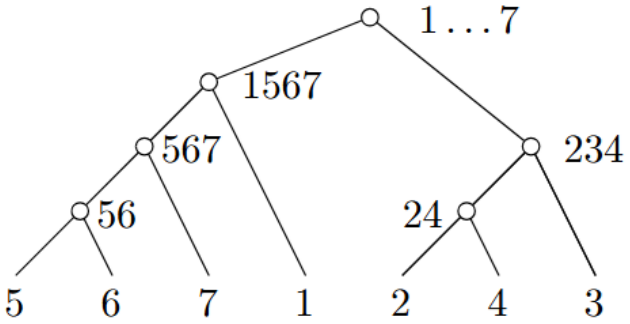


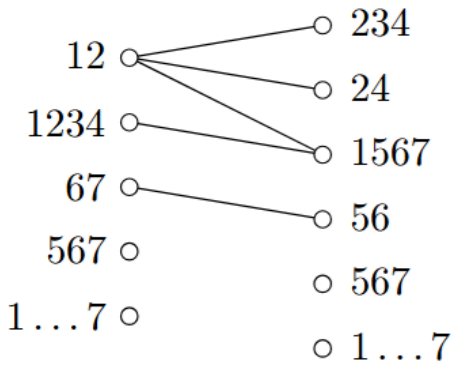
Incompatibility graphs

Let T_1, T_2 be two trees. Then $G(T_1, T_2)$ is the incompatibility graph of T_1, T_2. It has:

- one vertex for each cluster of T_1 and each cluster of T_2;
- an edge between clusters X and Y if and only if X, Y are incompatible.







Some motivations

Incompatibility graphs are notably used to:

- Reconstruct phylogenetic networks to combine trees with incompatible clusters. Each connected component of $G(T_1, T_2)$ corresponds to a blob of the network. [van Iersel et al, 2010]

Some motivations

Incompatibility graphs are notably used to:

- Reconstruct phylogenetic networks to combine trees with incompatible clusters. Each connected component of $G(T_1, T_2)$ corresponds to a blob of the network. [van Iersel et al, 2010]
- Compute distances between trees. The BHV distance projects the trees in a continuous vector space, and asks for the shortest geodesic path between the projections.

  A maximum independent set of $G(T_1, T_2)$ gives the direction to take in a shortest path. [Owen and Provan, 2010]

Some motivations

Incompatibility graphs are notably used to:

- Reconstruct phylogenetic networks to combine trees with incompatible clusters. Each connected component of $G(T_1, T_2)$ corresponds to a blob of the network. [van Iersel et al, 2010]
- Compute distances between trees. The BHV distance projects the trees in a continuous vector space, and asks for the shortest geodesic path between the projections.

  A maximum independent set of $G(T_1, T_2)$ gives the direction to take in a shortest path. [Owen and Provan, 2010]
- others (e.g., obtain consensus trees)

Computing $G(T_1, T_2)$

$n =$ number of leaves of T_1 and of T_2.

Current implementations

- Represent clusters as bit vectors. Operations take time $O(n/w)$, where w is the machine word size.
- Then **for each pair** of clusters, check for compatibility in time $O(n/w)$.
- Total time: $O(n^2 \cdot n/w) = O(n^3/w)$.

Computing $G(T_1, T_2)$

$O(n^2)$ seems to be a lower bound on the complexity, because incompatibility graphs can have $\Theta(n^2)$ edges.

Need quadratic time just to write the output.

Yes but...

This work

Theorem

Let T_1, T_2 be two trees on n leaves, and let $d = |E\big(G(T_1, T_2)\big)|$.

Then there is an algorithm that, on input T_1, T_2, outputs $G(T_1, T_2)$ in optimal time $O(n + d)$.

This work

Theorem

Let T_1, T_2 be two trees on n leaves, and let $d = |E\big(G(T_1, T_2)\big)|$.

Then there is an algorithm that, on input T_1, T_2, outputs $G(T_1, T_2)$ in optimal time $O(n + d)$.

Note: d is not part of the input.

An algorithm whose time complexity depends on the size of the output is called *output-sensitive*.

Similar result exist to enumerate conflicting rooted triples [Weller, 2018].

The algorithm

lca-maps

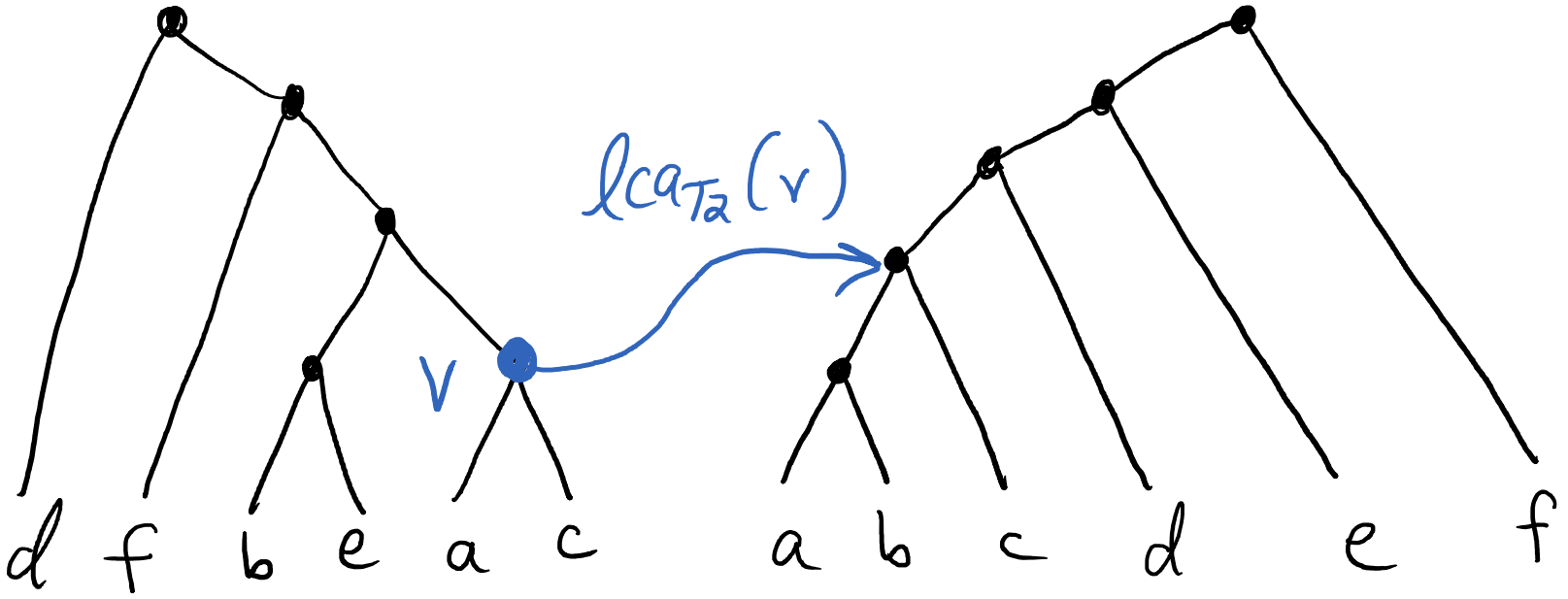
For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.

lca-maps

For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

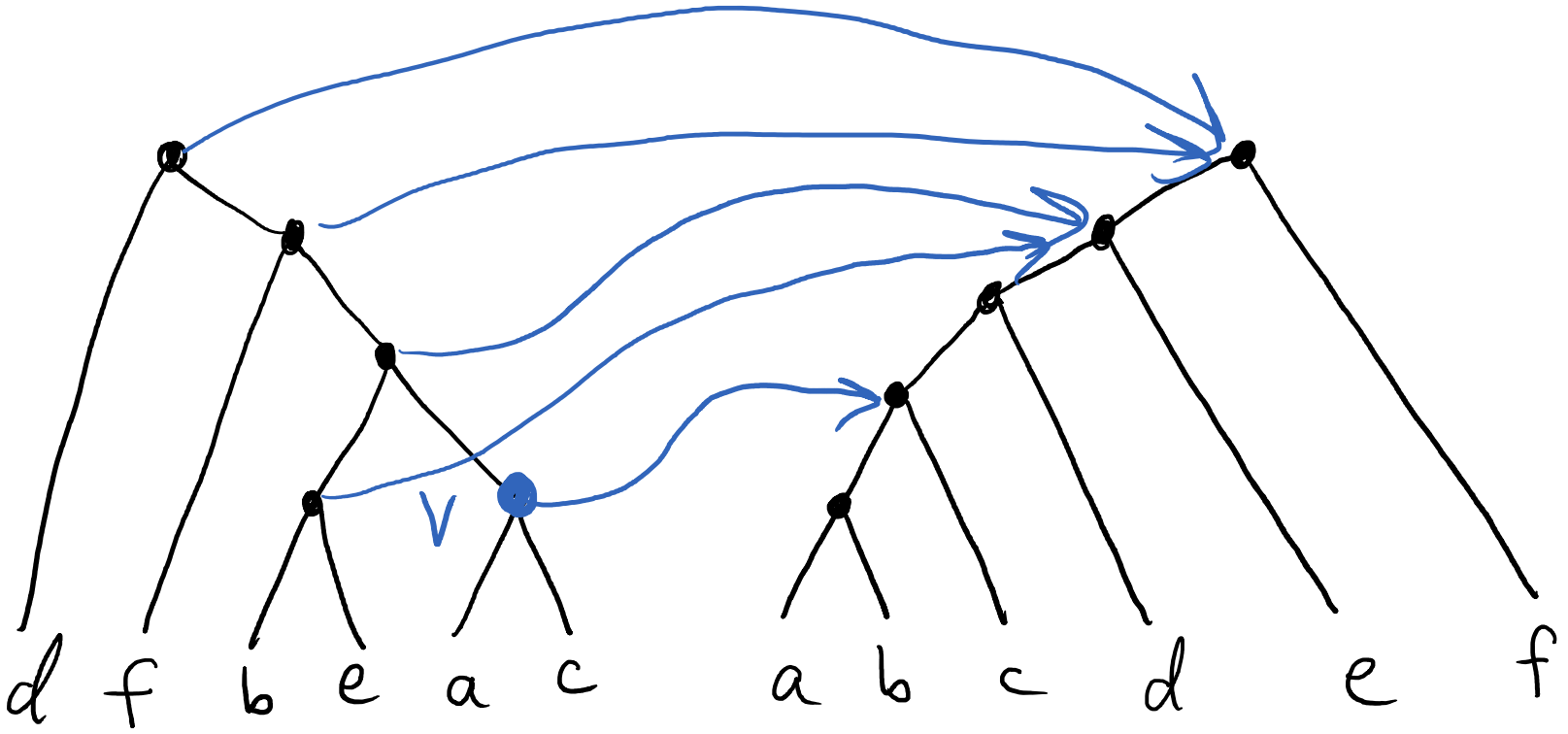
For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.



lca-maps

For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

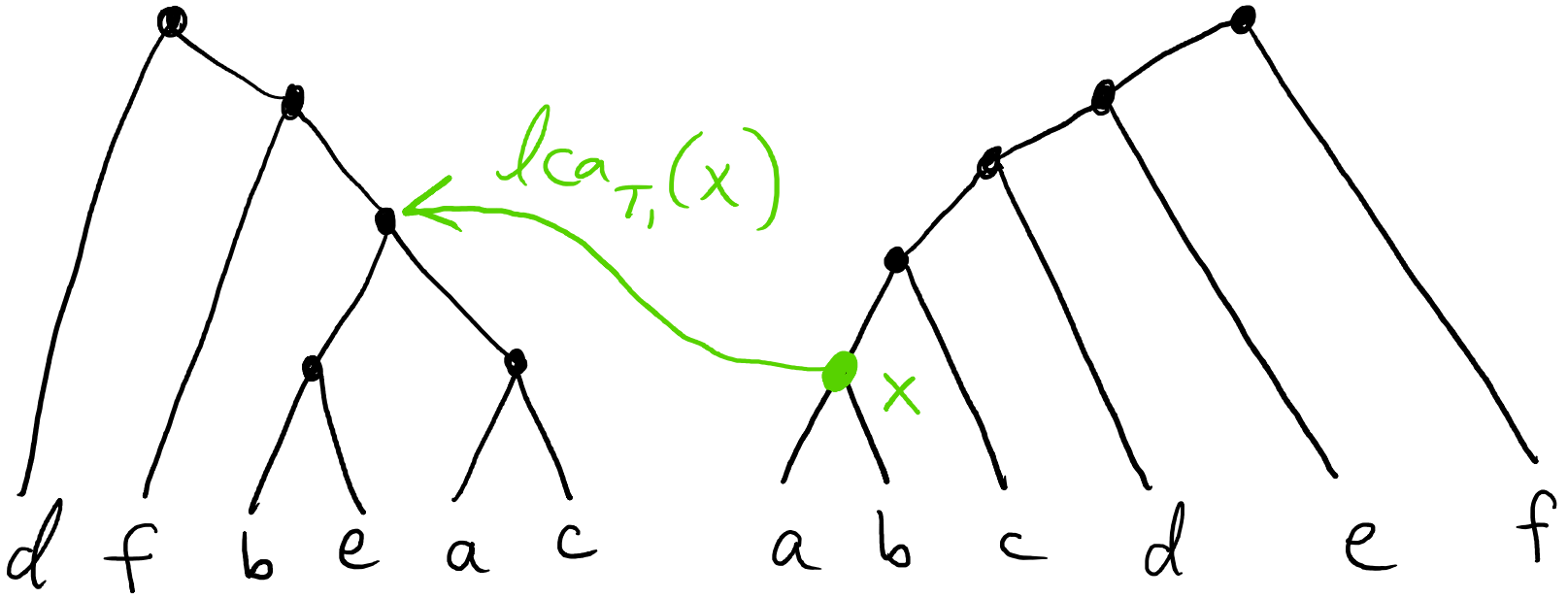
For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.



lca-maps

For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

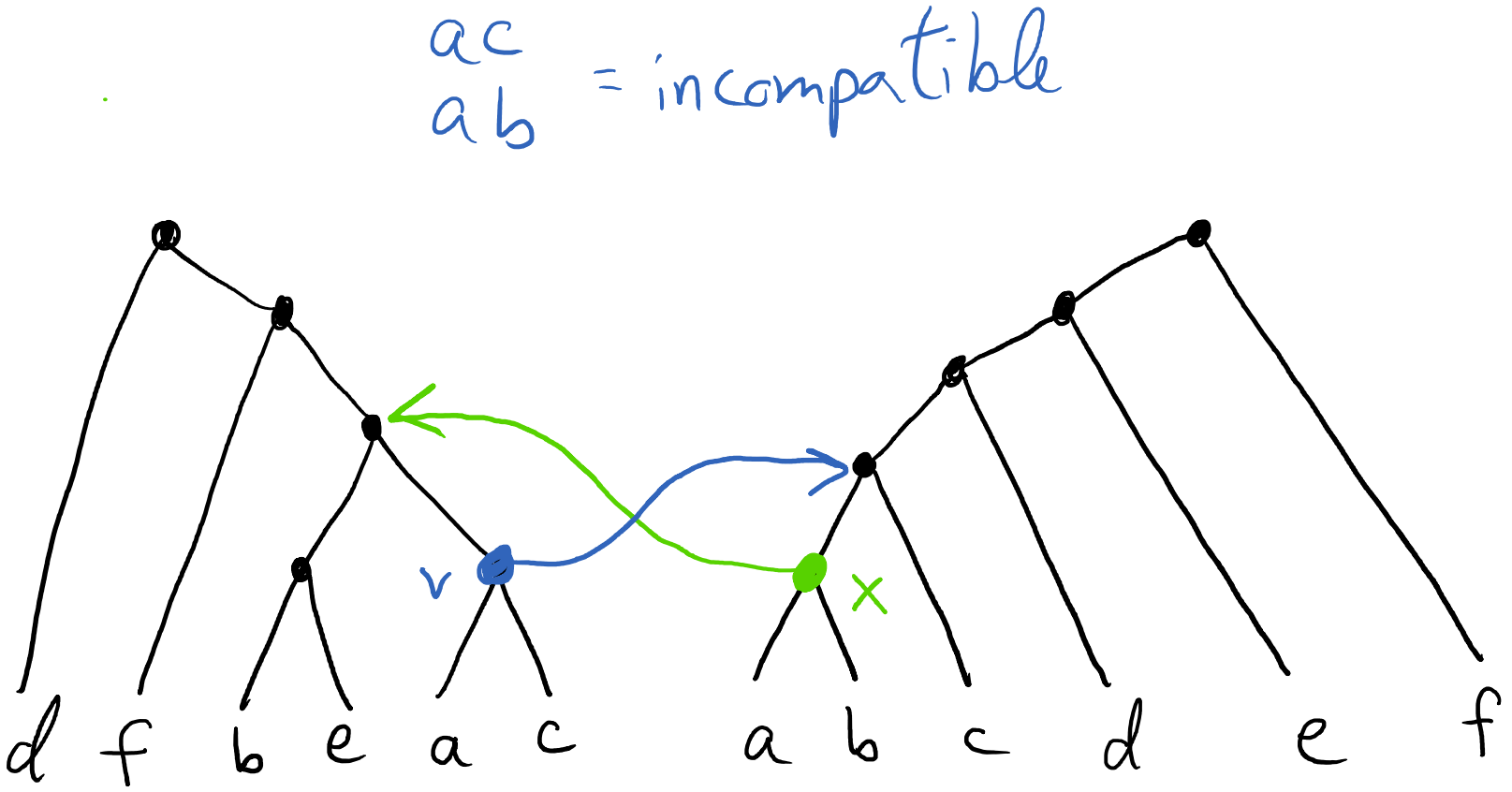
For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.



lca-maps

For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.



lca-maps

For $v \in V(T_1)$, $lca_{T_2}(v)$ is the lowest common ancestor of $cluster_{T_1}(v)$.

For $x \in V(T_2)$, $lca_{T_1}(x)$ is the lowest common ancestor of $cluster_{T_2}(x)$.

Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

Lemma

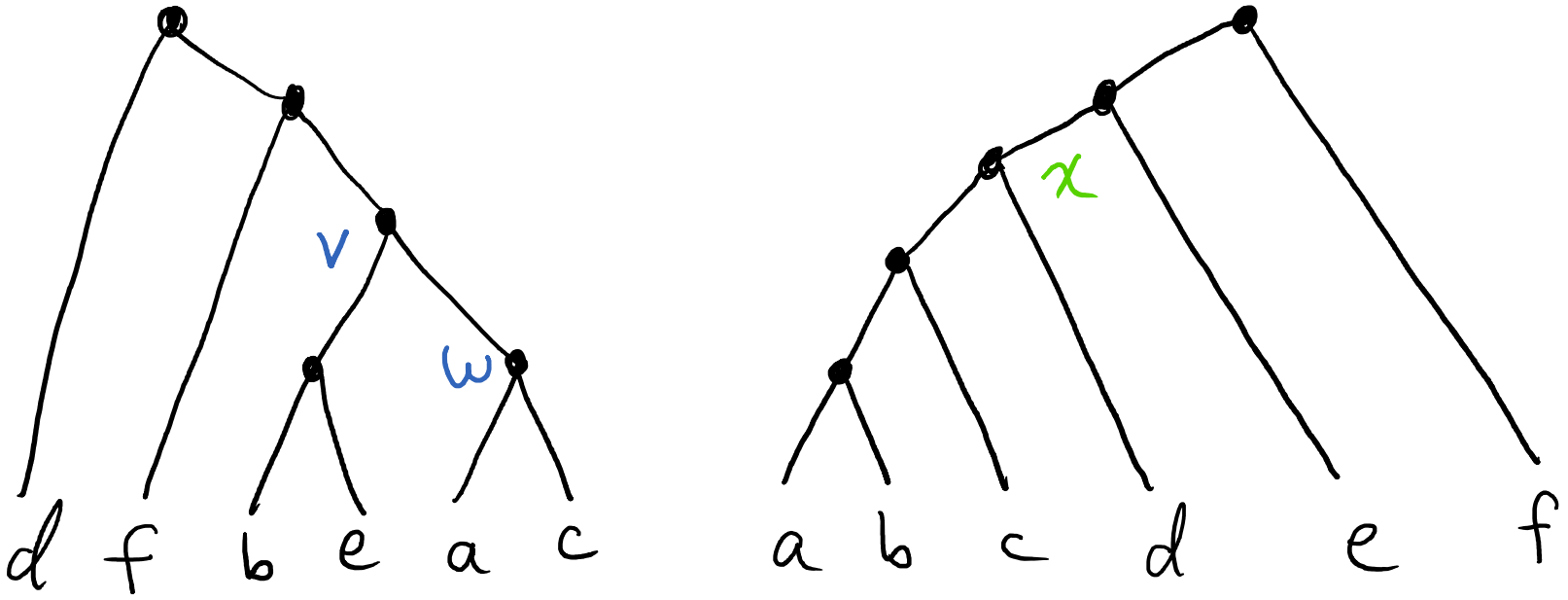
The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

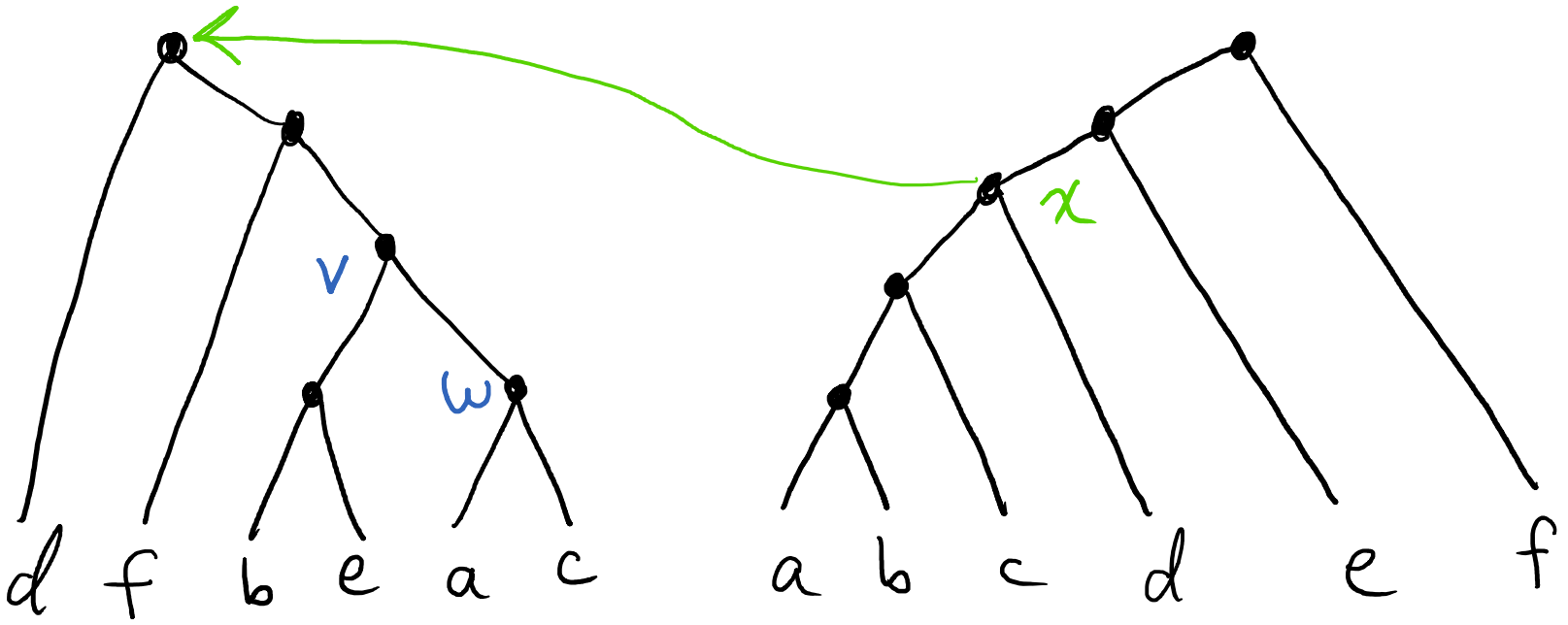


Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

$C(x) \not\subseteq C(v)$

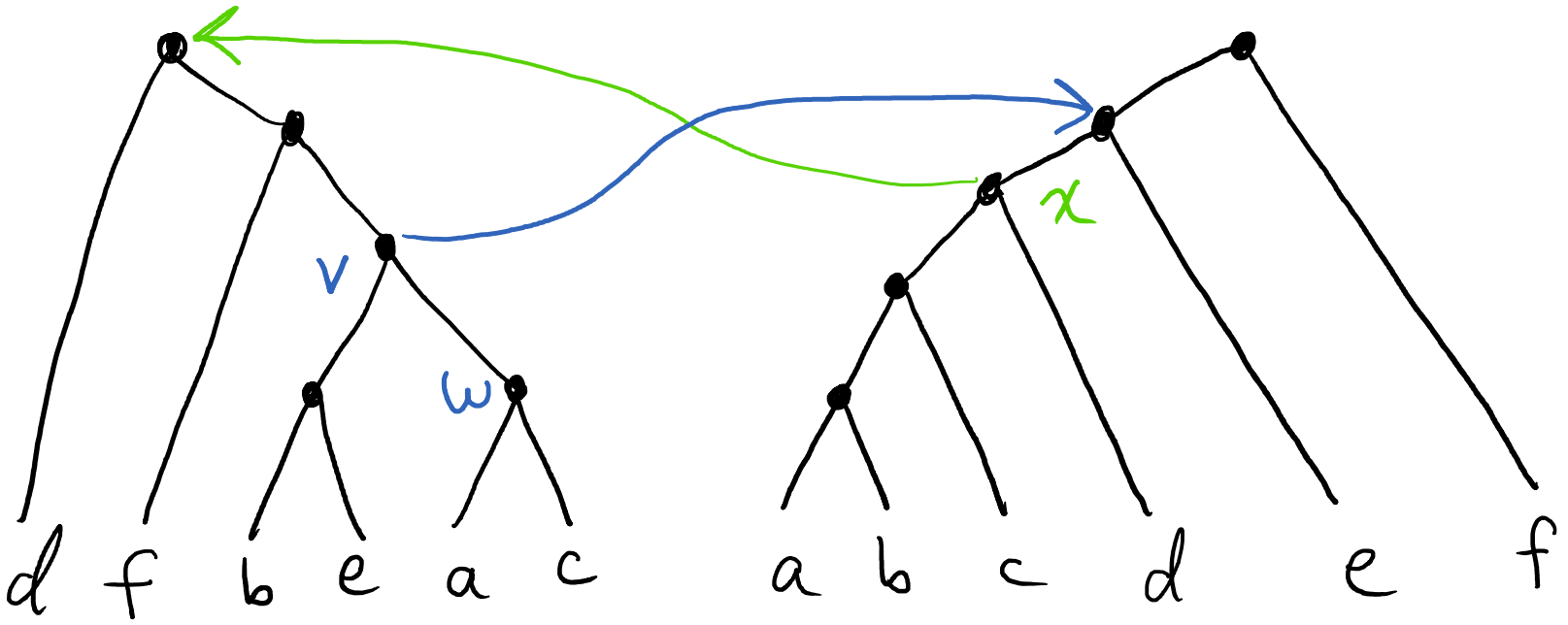


Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

$C(x) \not\subseteq C(v)$ $\quad$ $C(v) \not\subseteq C(x)$



Lemma

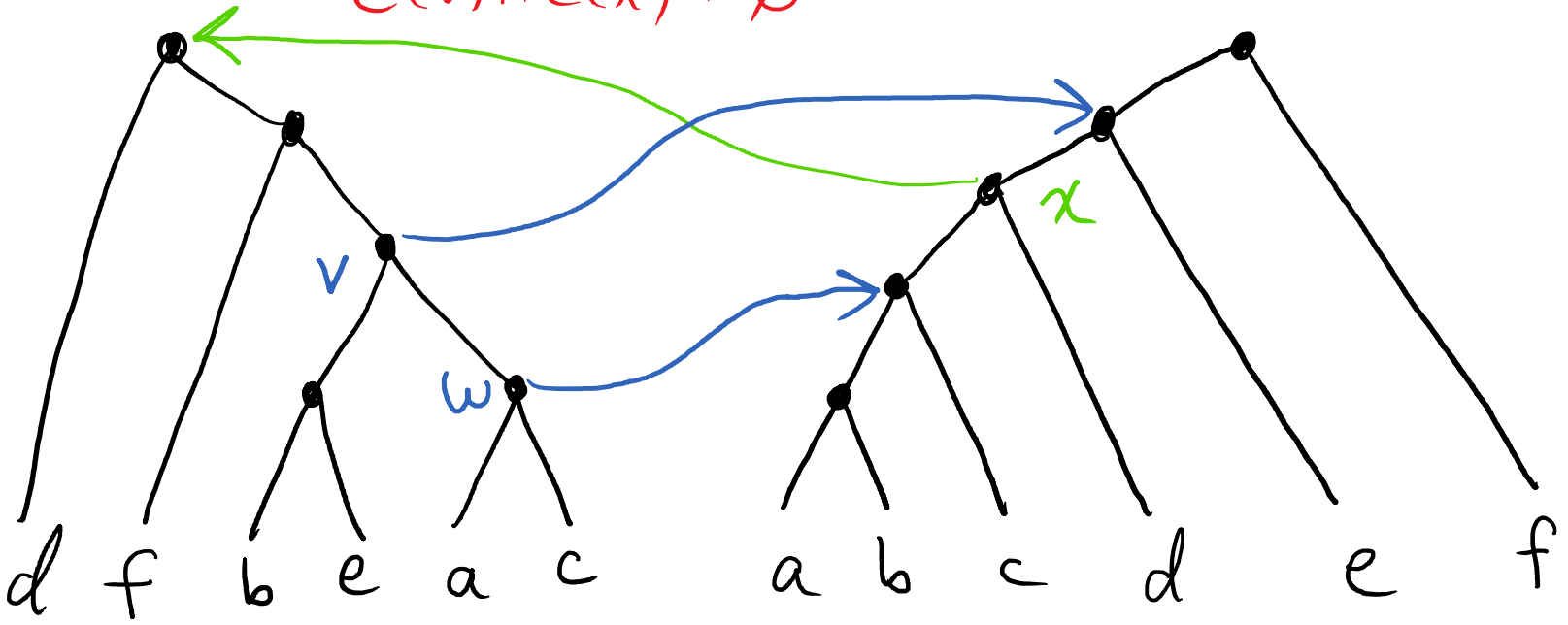
The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

$C(x) \not\subseteq C(v)$

$C(v) \not\subseteq C(x)$

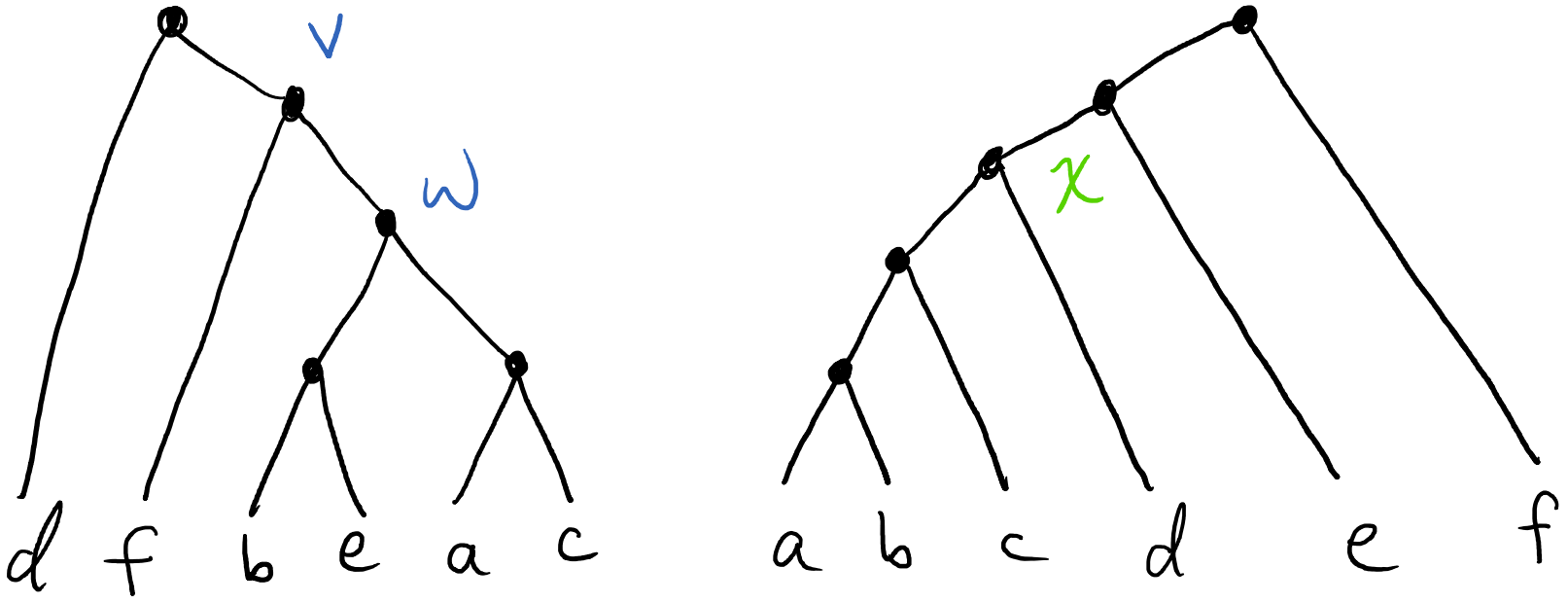
$C(v) \cap C(x) \neq \emptyset$



Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

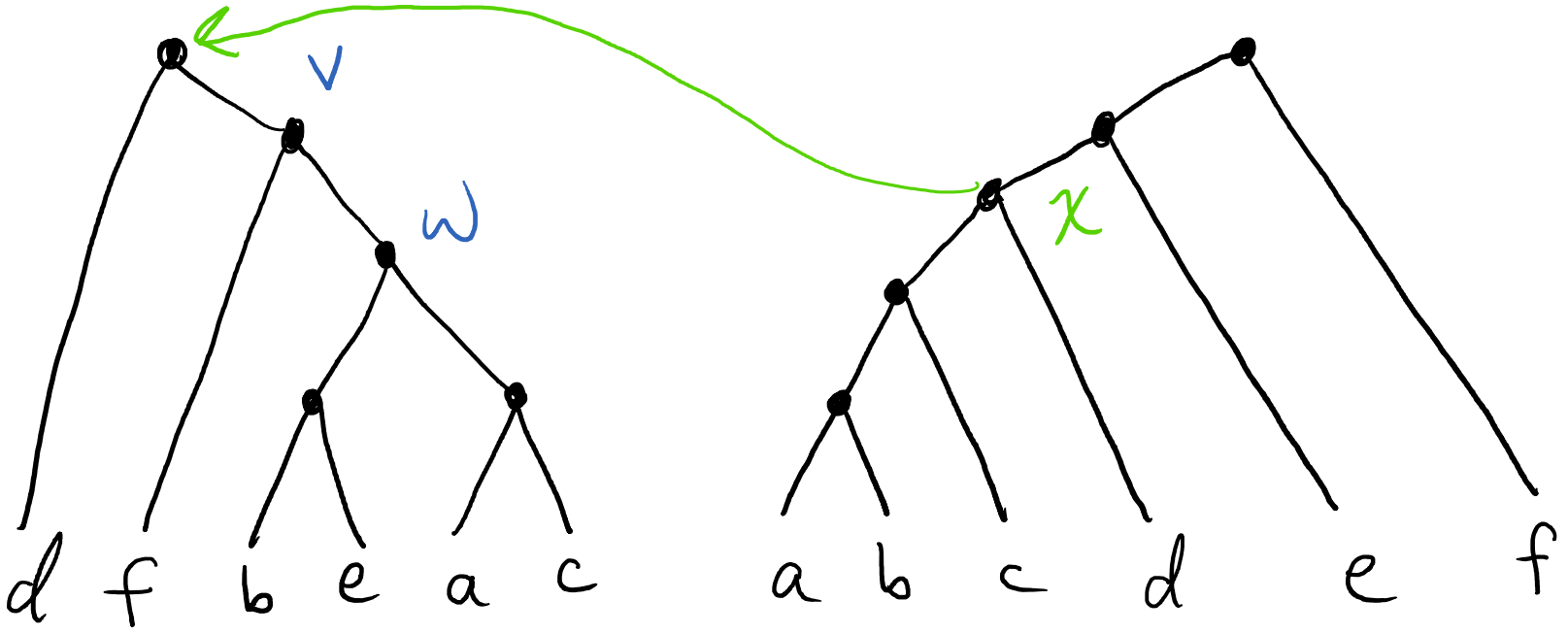


Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

$C(x) \not\subseteq C(v)$



Lemma

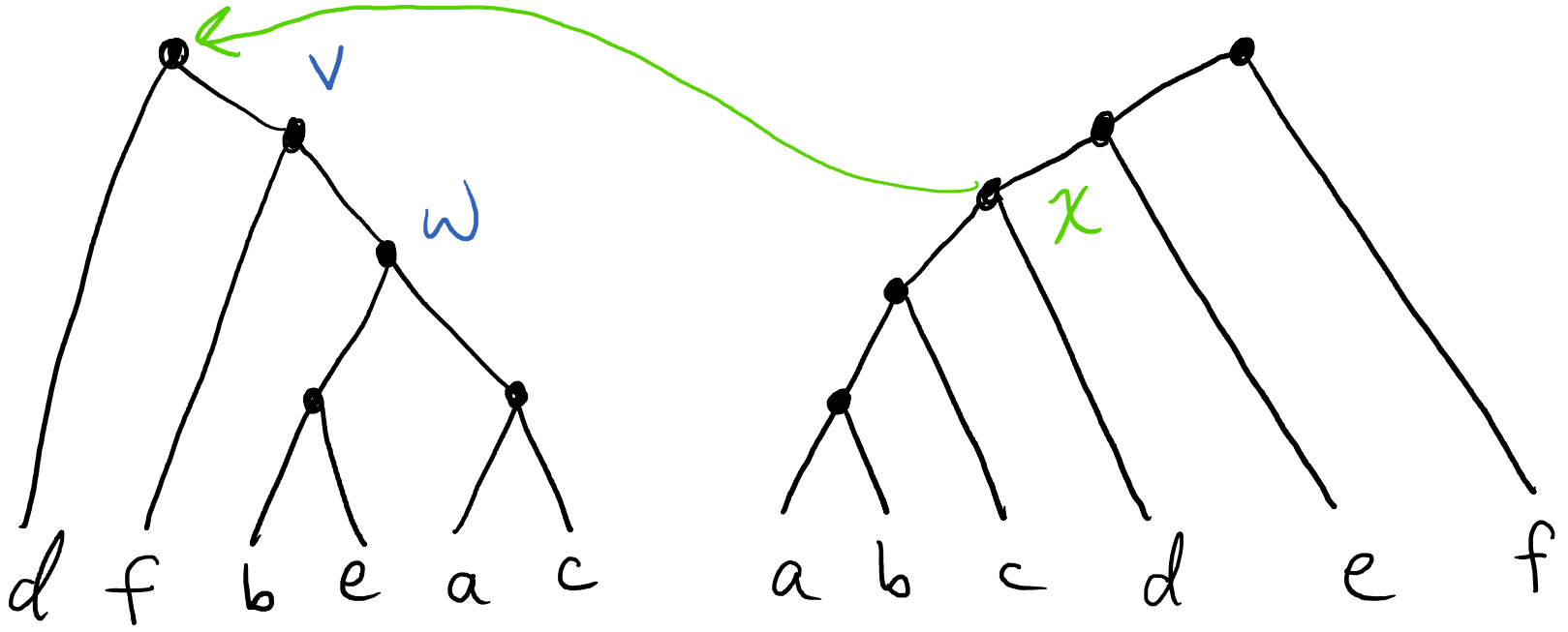
The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

$C(x) \not\subseteq C(v)$

$C(w) \not\subseteq C(x) \Rightarrow C(v) \not\subseteq C(x)$

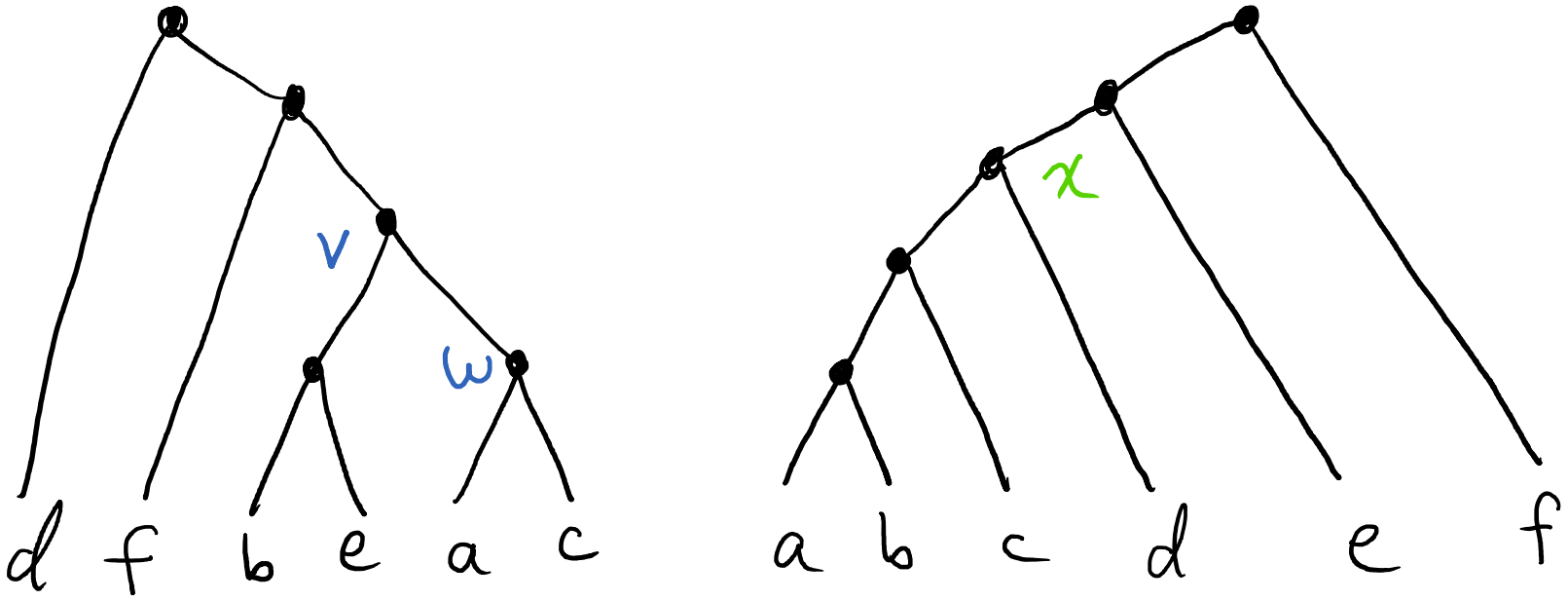
$C(w) \cap C(x) \neq \emptyset \Rightarrow C(v) \cap C(x) \neq \emptyset$



Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if at least one of the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.



```
1  function build_incompatibility_graph(T, T')
2      Pre-process T, T' for O(1) lca queries
3      Construct the lca-maps μ⃗ and ν⃖
4      Pre-process T' for O(1) diff_anc(x) queries
5      Let G be a graph with vertex set V(T) ∪ V(T') and no edge
6      Let Q be a global array of length |V(T')|, with values initialized to −1
7      for each v ∈ I(T) in a postorder traversal do
8          check_between(v, G)
9          check_incomp_child(v, G)
10
11 function check_between(v, G)
12     for each child w of v do
13         x = μ⃗(w)
14         while x ≠ ⊥ and ν⃖(x) ⪯_T v do
15             x = diff_anc(x)
16         while x ≠ ⊥ and x ≺_T' μ⃗(v) and Q[x] ≠ v do
17             Add {v, x} to E(G)
18             Q[x] = v
19             x = parent_T'(x)
20
21 function check_incomp_child(v, G)
22     for each child w of v do
23         for each edge {w, x} incident to w in G do
24             if v ≺_T ν⃖(x) and Q[x] ≠ v then
25                 Add {v, x} to E(G)
26                 Q[x] = v
```

```
1  function build_incompatibility_graph(T, T')
2      Pre-process T, T' for O(1) lca queries
3      Construct the lca-maps μ⃗ and ν⃖
4      Pre-process T' for O(1) diff_anc(x) queries
5      Let G be a graph with vertex set V(T) ∪ V(T') and no edge
6      Let Q be a global array of length |V(T')|, with values initialized to −1
7      for each v ∈ I(T) in a postorder traversal do
8          check_between(v, G)
9          check_incomp_child(v, G)
10
11 function check_between(v, G)
12     for each child w of v do
13         x = μ⃗(w)
14         while x ≠ ⊥ and ν⃖(x) ⪯_T v do
15             x = diff_anc(x)
16         while x ≠ ⊥ and x ≺_T' μ⃗(v) and Q[x] ≠ v do
17             Add {v, x} to E(G)
18             Q[x] = v
19             x = parent_T'(x)
20
21 function check_incomp_child(v, G)
22     for each child w of v do
23         for each edge {w, x} incident to w in G do
24             if v ≺_T ν⃖(x) and Q[x] ≠ v then
25                 Add {v, x} to E(G)
26                 Q[x] = v
```

Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

for each $v \in V(T_1)$:

    for each child w of v:

        for each edge $\{w, x\}$:

            if $v \prec lca_{T_1}(x)$, add $\{v, x\}$ to $G(T_1, T_2)$

Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

for each $v \in V(T_1)$:
    for each child w of v:
        for each edge $\{w, x\}$:
            if $v \prec lca_{T_1}(x)$, add $\{v, x\}$ to $G(T_1, T_2)$

costs total $O(n)$

costs total $O(d)$

Lemma

The clusters of nodes $v \in V(T_1)$ and $x \in V(T_2)$ are incompatible if and only if the two following conditions is true:

- $v \prec lca_{T_1}(x)$ and v has a child w such that $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$; or
- $v \prec lca_{T_1}(x)$ and v has a child w such that w and x are incompatible.

for each $v \in V(T_1)$:

    for each child w of v:

        $x \leftarrow$ first ancestor of $lca_{T_2}(w)$ with $v \prec lca_{T_1}(x)$

        while $lca_{T_2}(w) \preccurlyeq x \prec lca_{T_2}(v)$

            add $\{v, w\}$ to $G(T_1, T_2)$

            $x \leftarrow x.parent$

Experiments

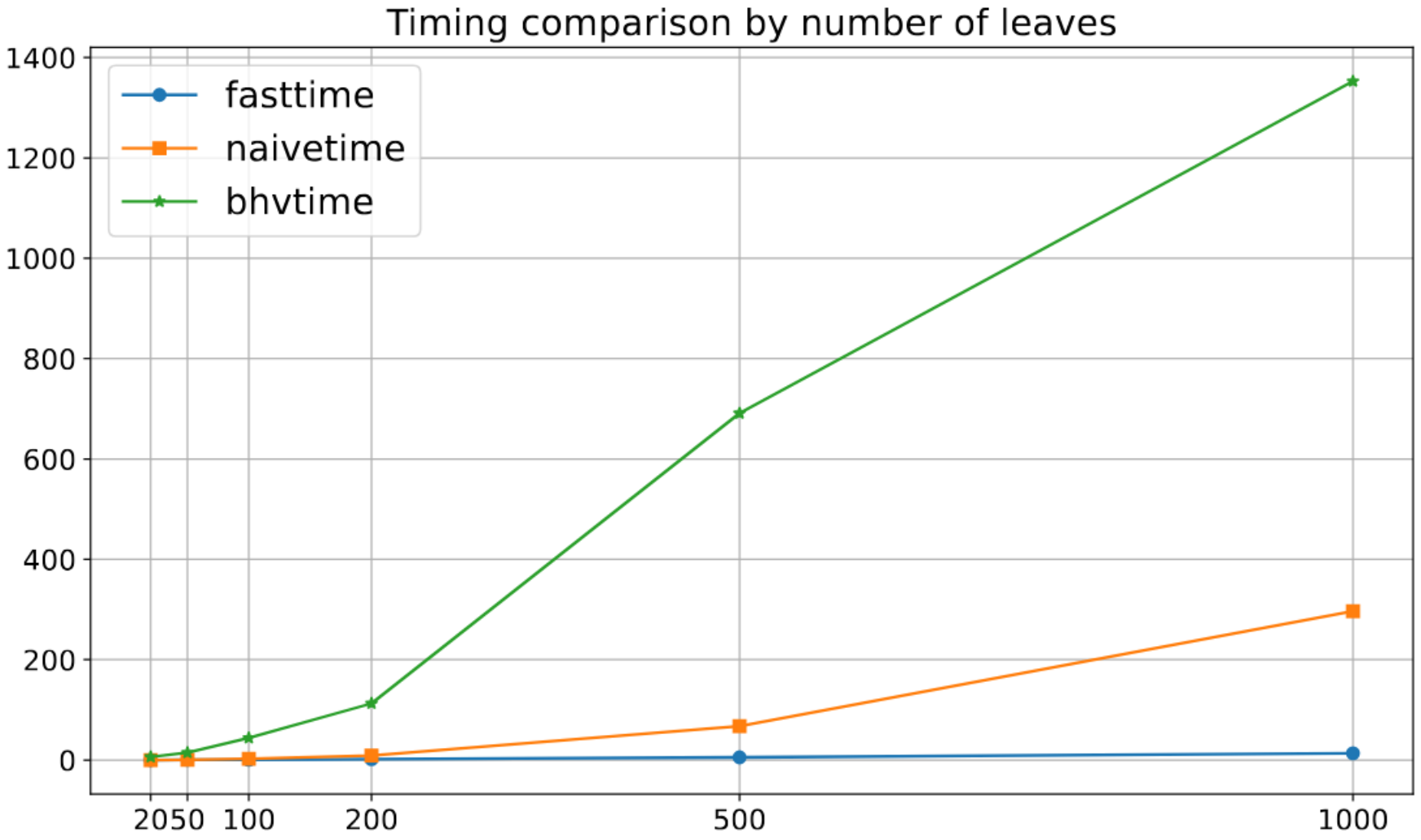
Two datasets

- Random trees (each chosen uniformly at random)
- SimPhy trees
  - Gene trees simulated with various levels of Incomplete Lineage Sorting (ILS)
  - More ILS = More dissimilar trees
- $n \in \{20, 50, 100, 200, 500, 1000\}$

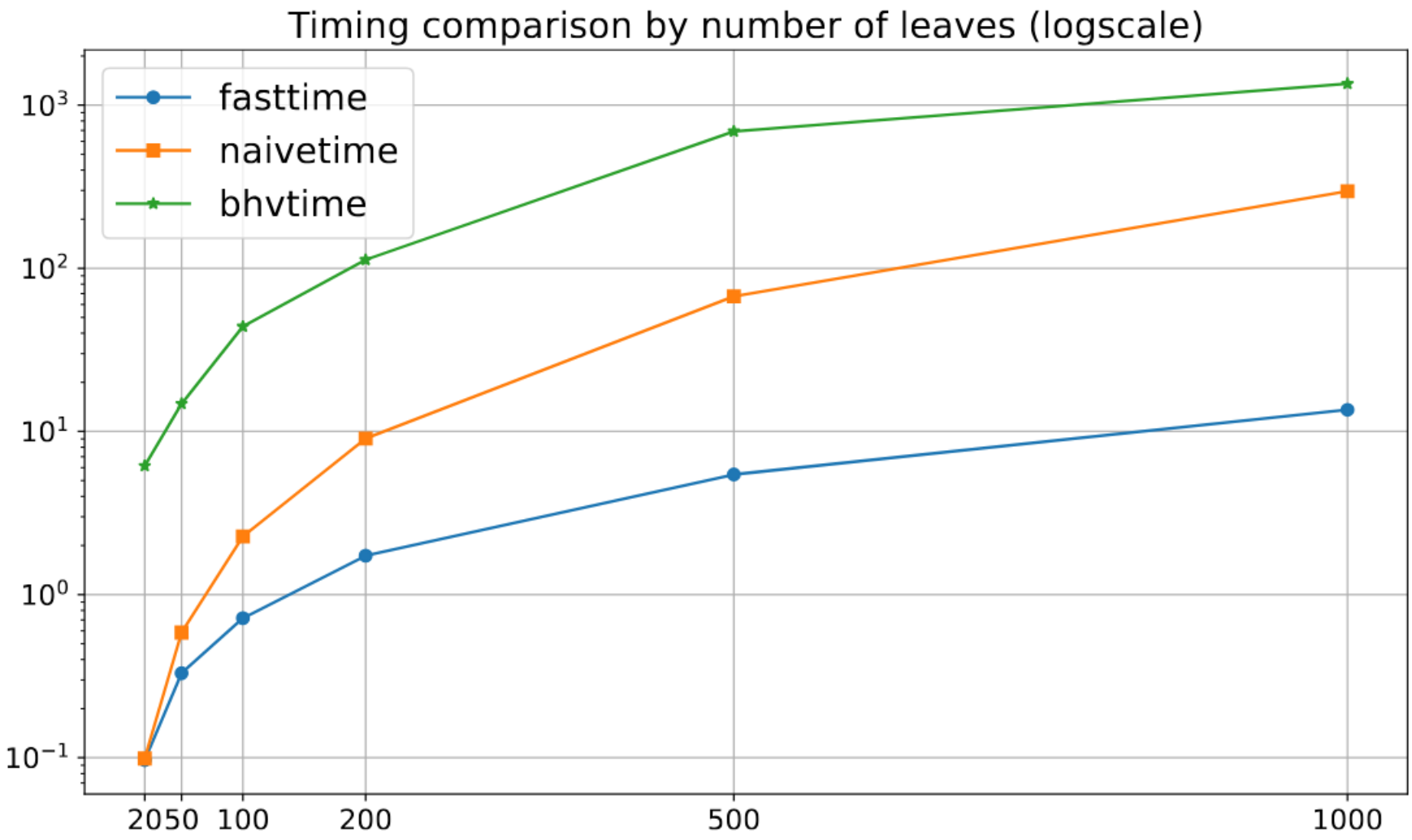
Three implementations

- Naive: the $O(n^3/w)$ algorithm (my C++ code)
- BHV: the $O(n^3/w)$ algorithm from the BHV distance (Owen & Provan's Java code)
- Fast: the $O(n + d)$ algorithm

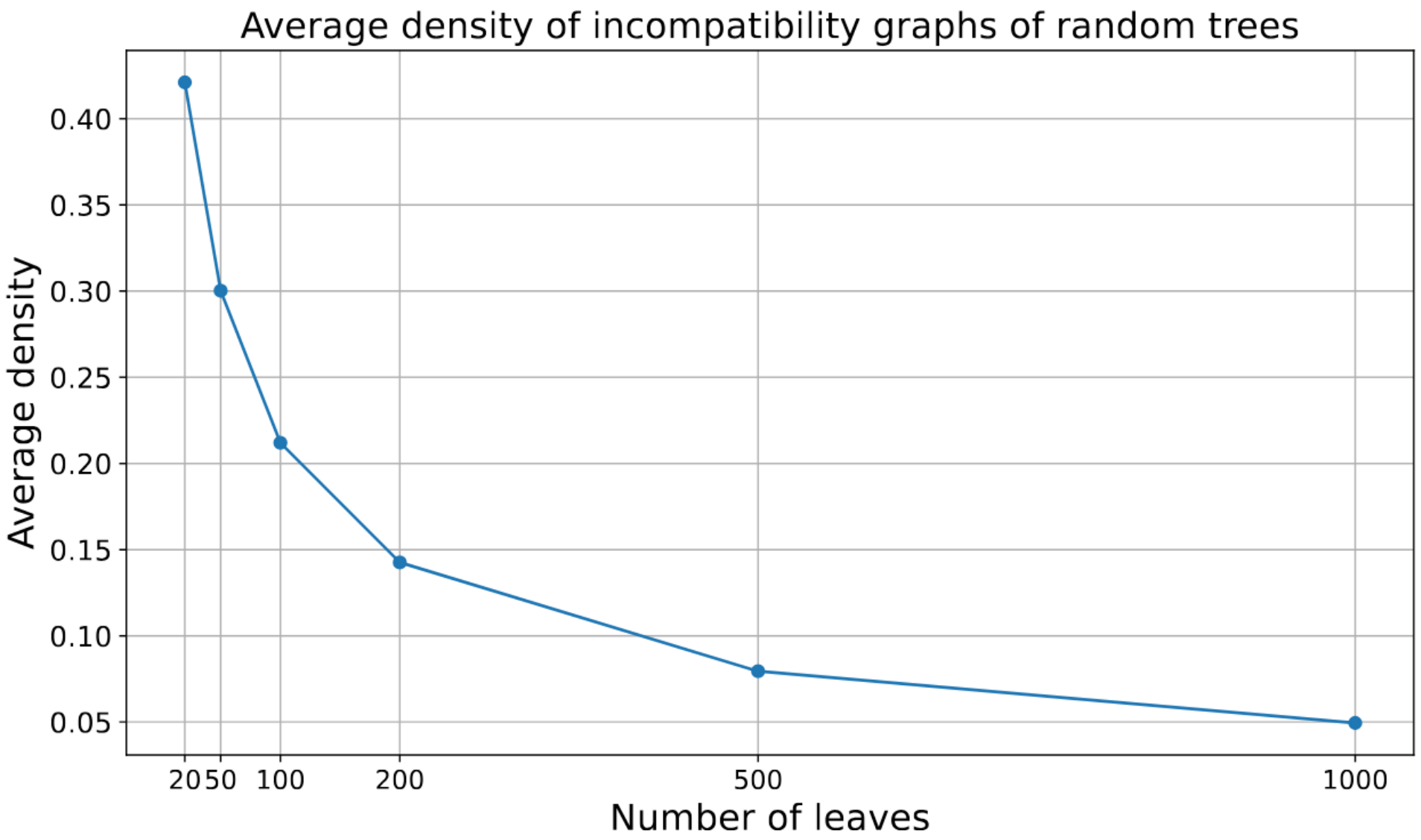
Running time on random trees



Running time on random trees



Density on random trees $\frac{|E(G(T_1,T_2))|}{|V(T_1)||V(T_2)|}$



Density on random trees $\frac{|E(G(T_1,T_2))|}{|V(T_1)||V(T_2)|}$

Density goes down as n grows.

$d = |E(G(T_1, T_2))|$ is closer to n on large trees.

Large trees are those that need fast algorithms!

- Good to have "close" to linear.

Density on random trees $\frac{|E(G(T_1,T_2))|}{|V(T_1)||V(T_2)|}$

Density goes down as n grows.

$d = |E(G(T_1, T_2))|$ is closer to n on large trees.

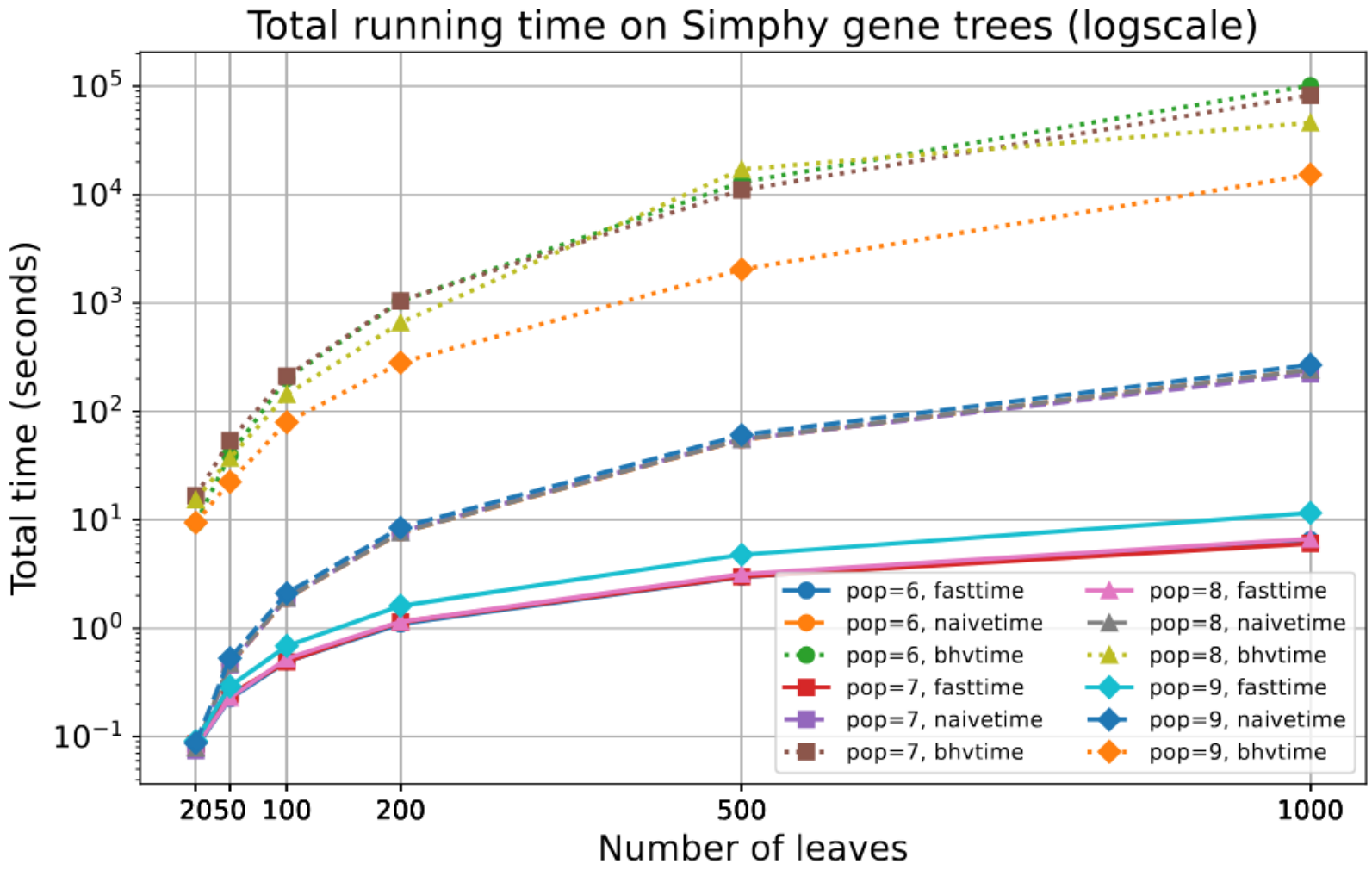
Large trees are those that need fast algorithms!

- Good to have "close" to linear.

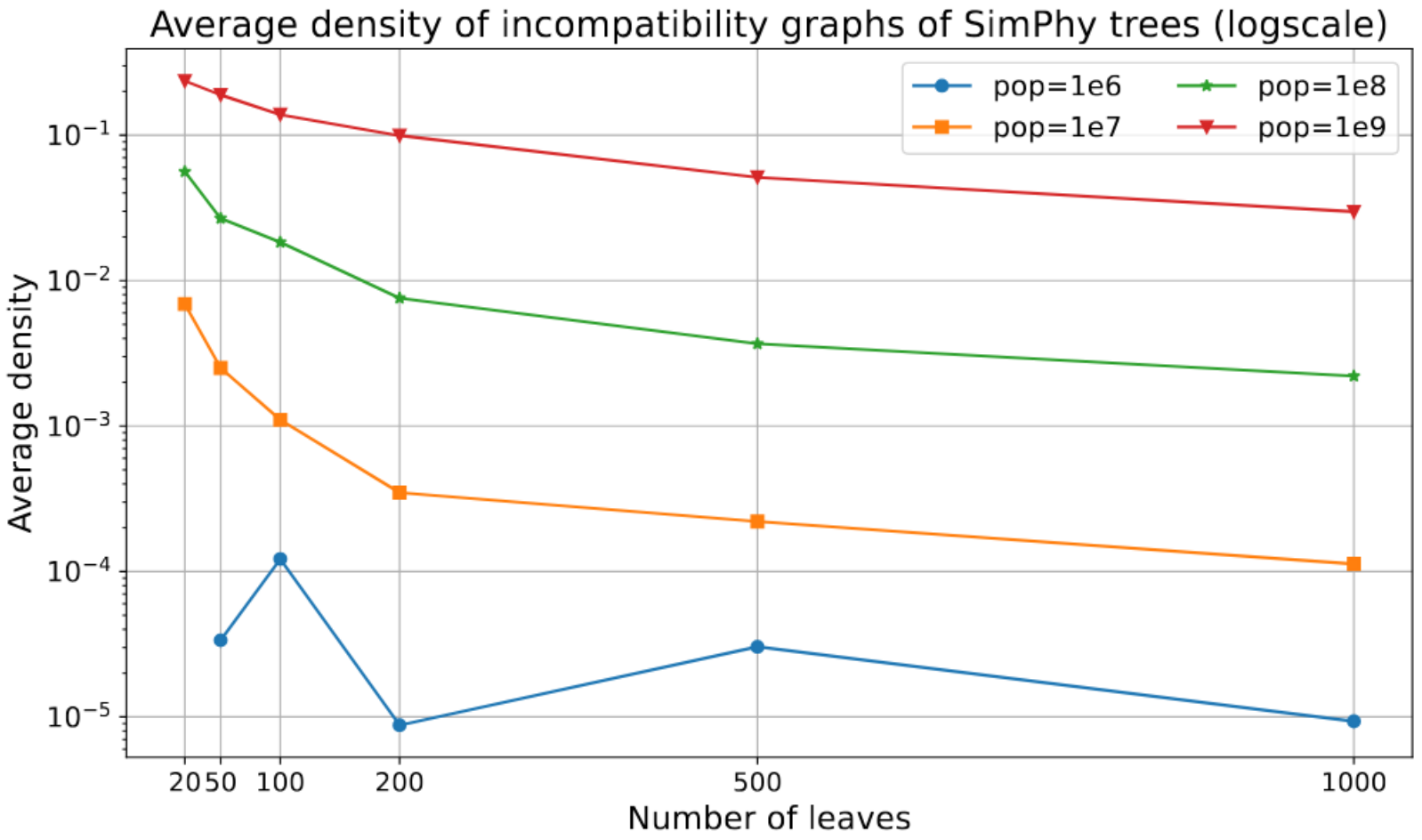
Why does density go down?

- Large trees have lots of tiny clusters, which are compatible with almost everything.

Running time on SimPhy trees



Density on SimPhy trees $\frac{|E(G(T_1,T_2))|}{|V(T_1)||V(T_2)|}$



Density on SimPhy trees $\frac{|E(G(T_1,T_2))|}{|V(T_1)||V(T_2)|}$

Density goes down, but also depends on rate of ILS.

More ILS = Denser incompatibility graphs.

Because more ILS = more differences.

Future work

1. Complexity of computing $G(T_1, T_2, \ldots, T_k)$?
   - Conjecture: $O(k^2 n + d)$ is optimal.
2. Using $|E\big(G(T_1, T_2)\big)|$ as a dissimilarity measure.
   - Are there nice metric properties? Triangle inequality?
   - Diameter with respect to tree shape
3. Better understand $G(T_1, T_2)$ on random trees.
   - Expected number of edges?