On the consistency of orthology relationships

Mark Jones Christophe Paul Céline Scornavacca

jones@lirmm.fr
LIRMM, Université de Montpellier

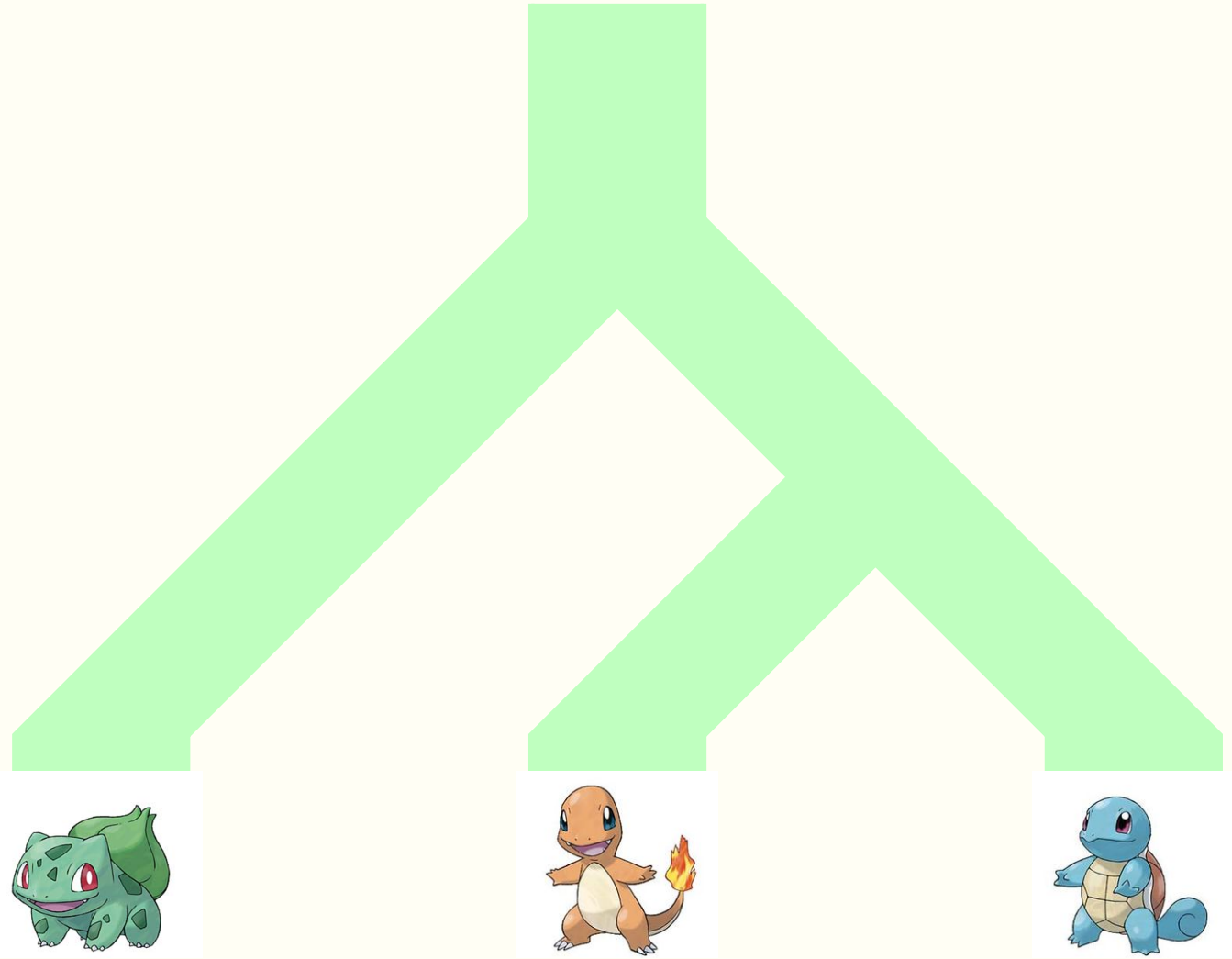
ALGCO seminar

Introduction

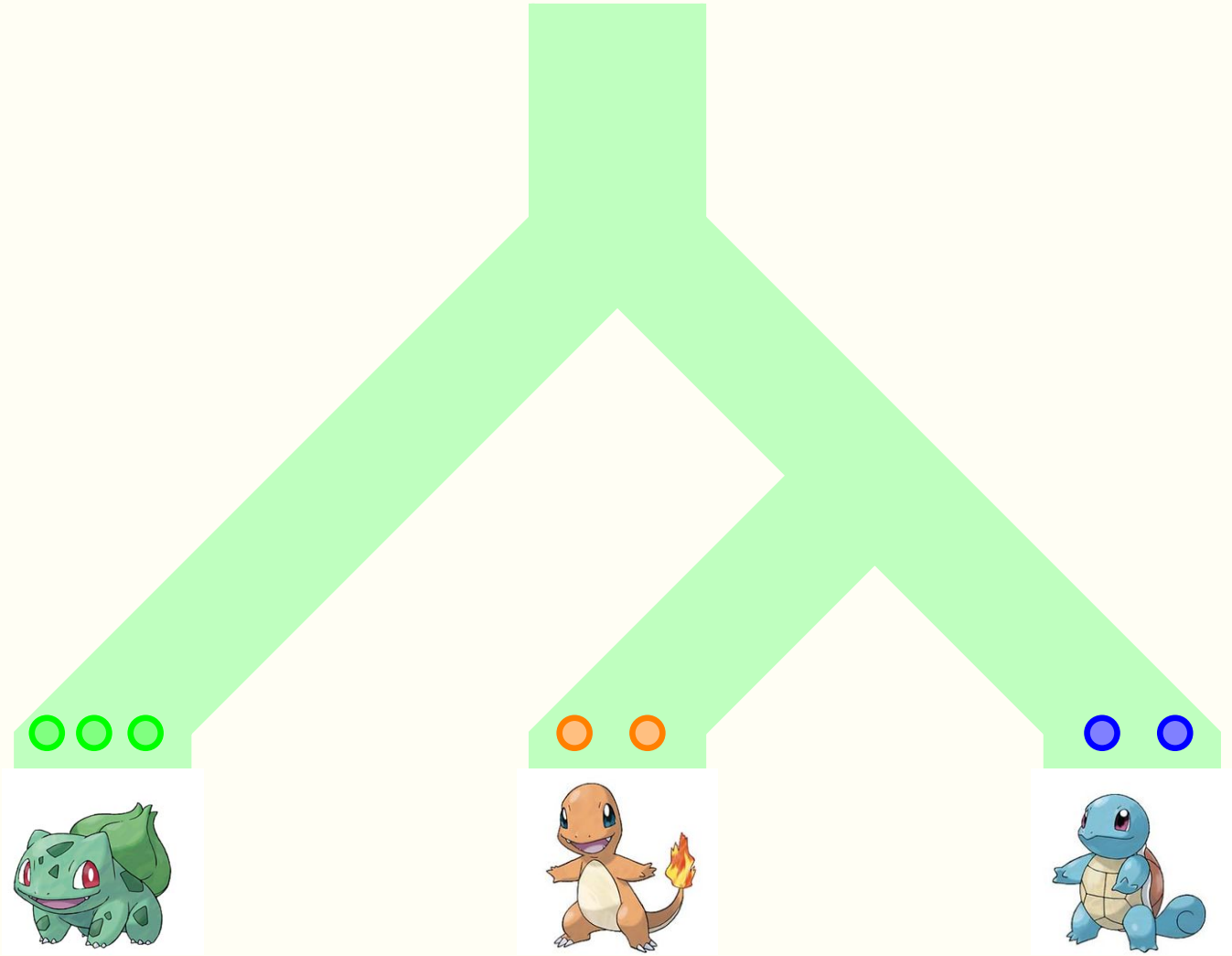
- Joint work with Christophe Paul(LIRMM) and Céline Scornavacca (Institut Des Sciences De L'volution De Montpellier).
- Orginally presented at RECOMB Comparative Genommics Workshop.
- Main result: A polynomial time algorithm for resolving orthology data between genes in a way that is consistent (i.e. can be made to fit with a species tree).

Evolution of Species vs Genes

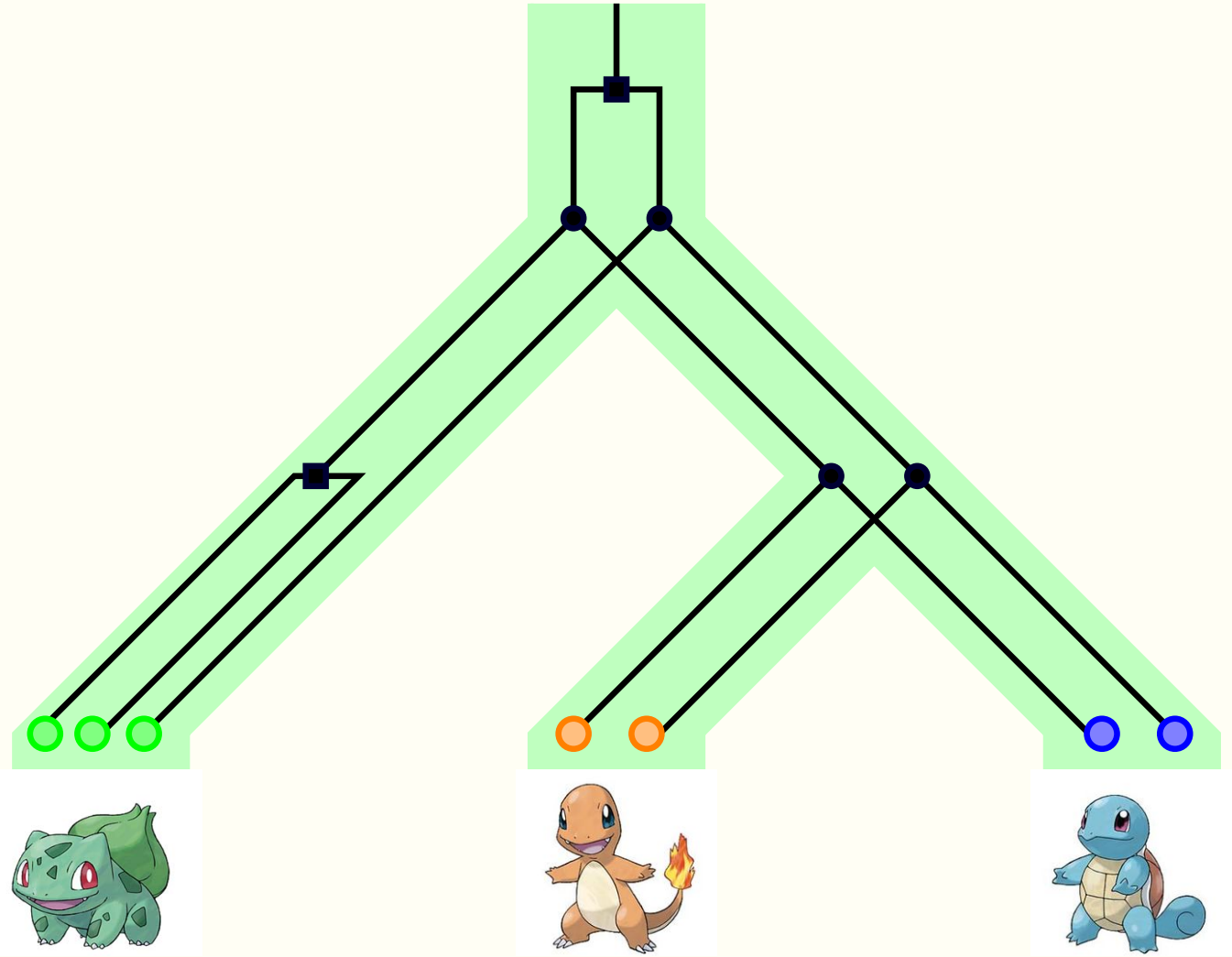
Evolution of Species vs Genes



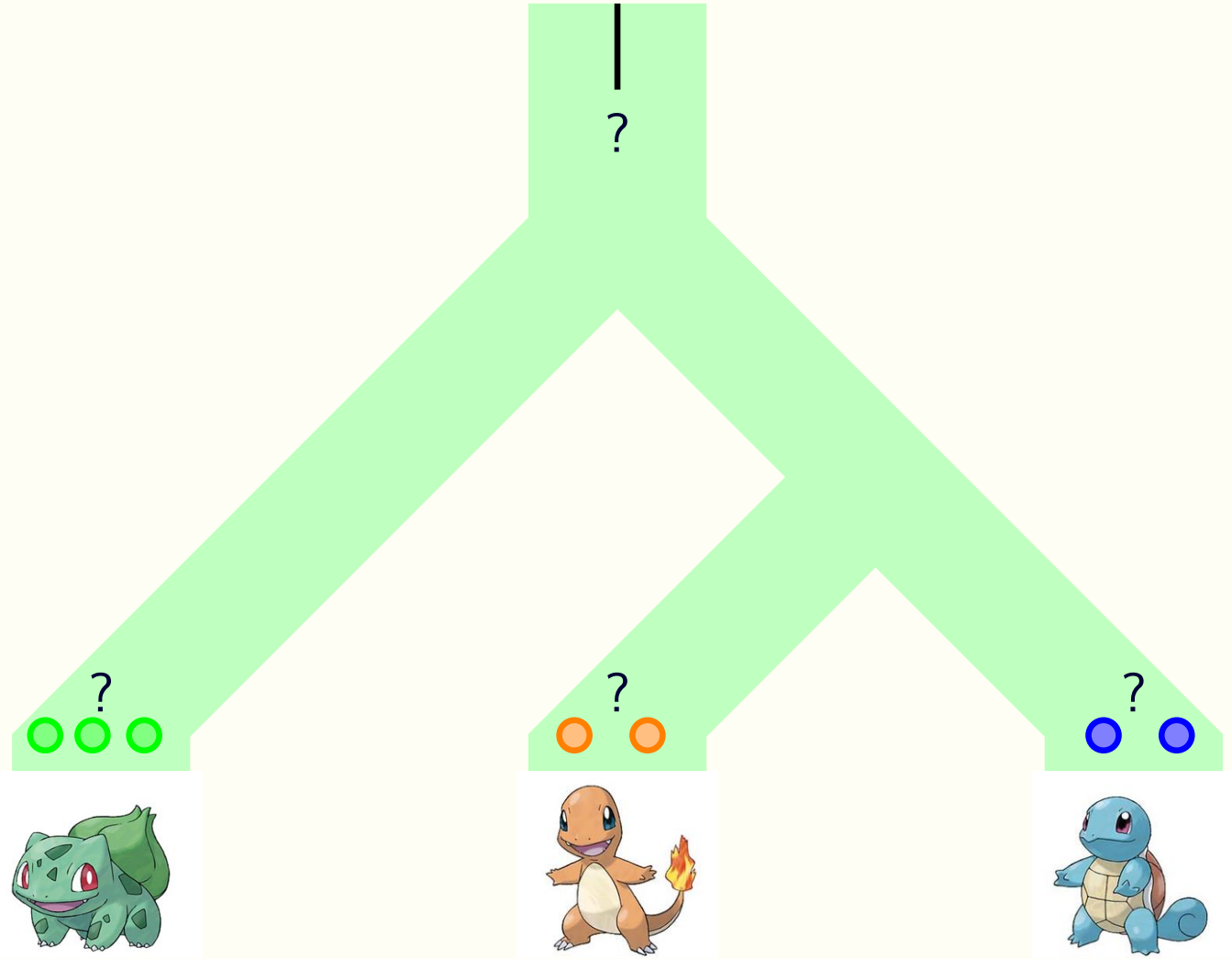
Evolution of Species vs Genes



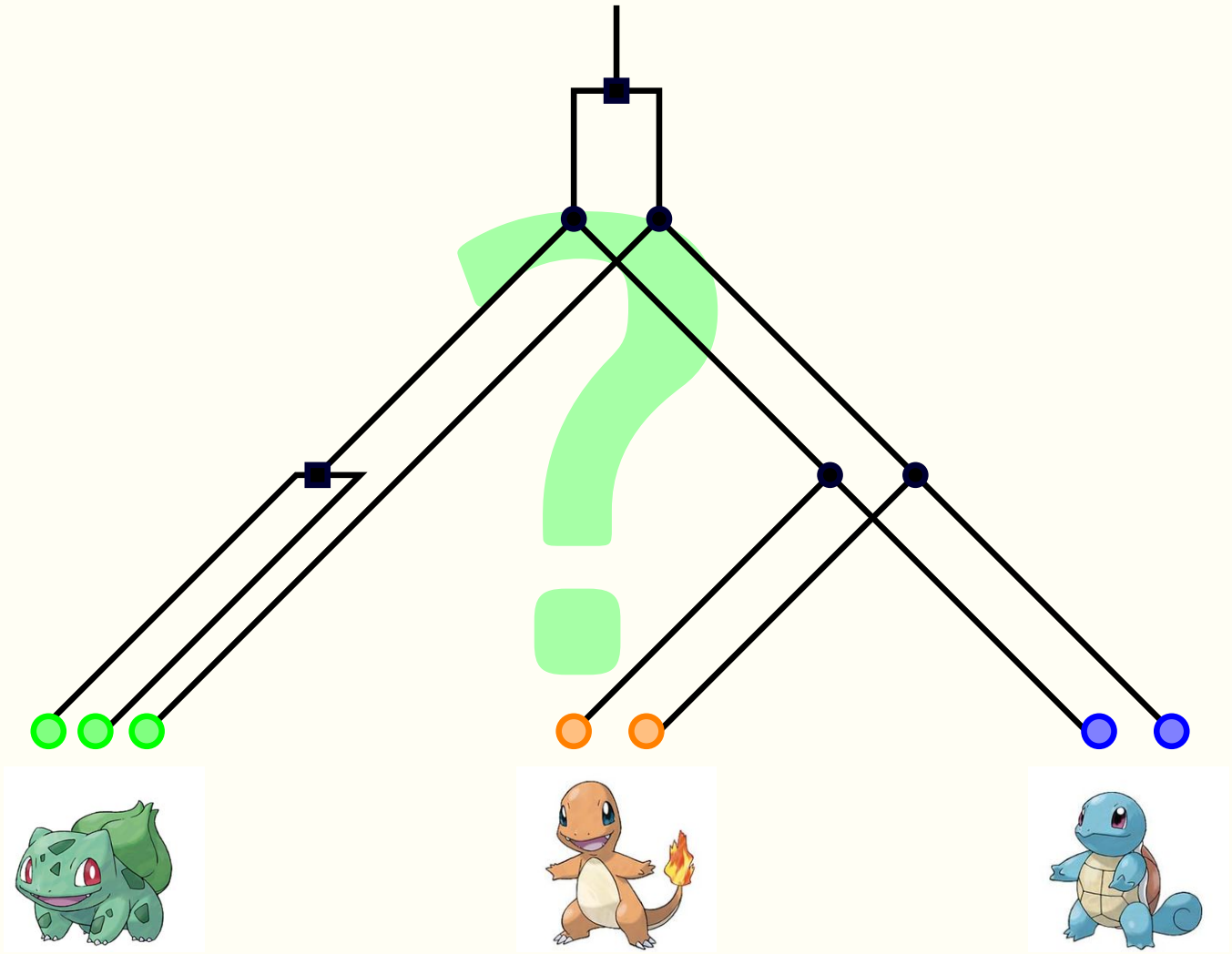
Evolution of Species vs Genes



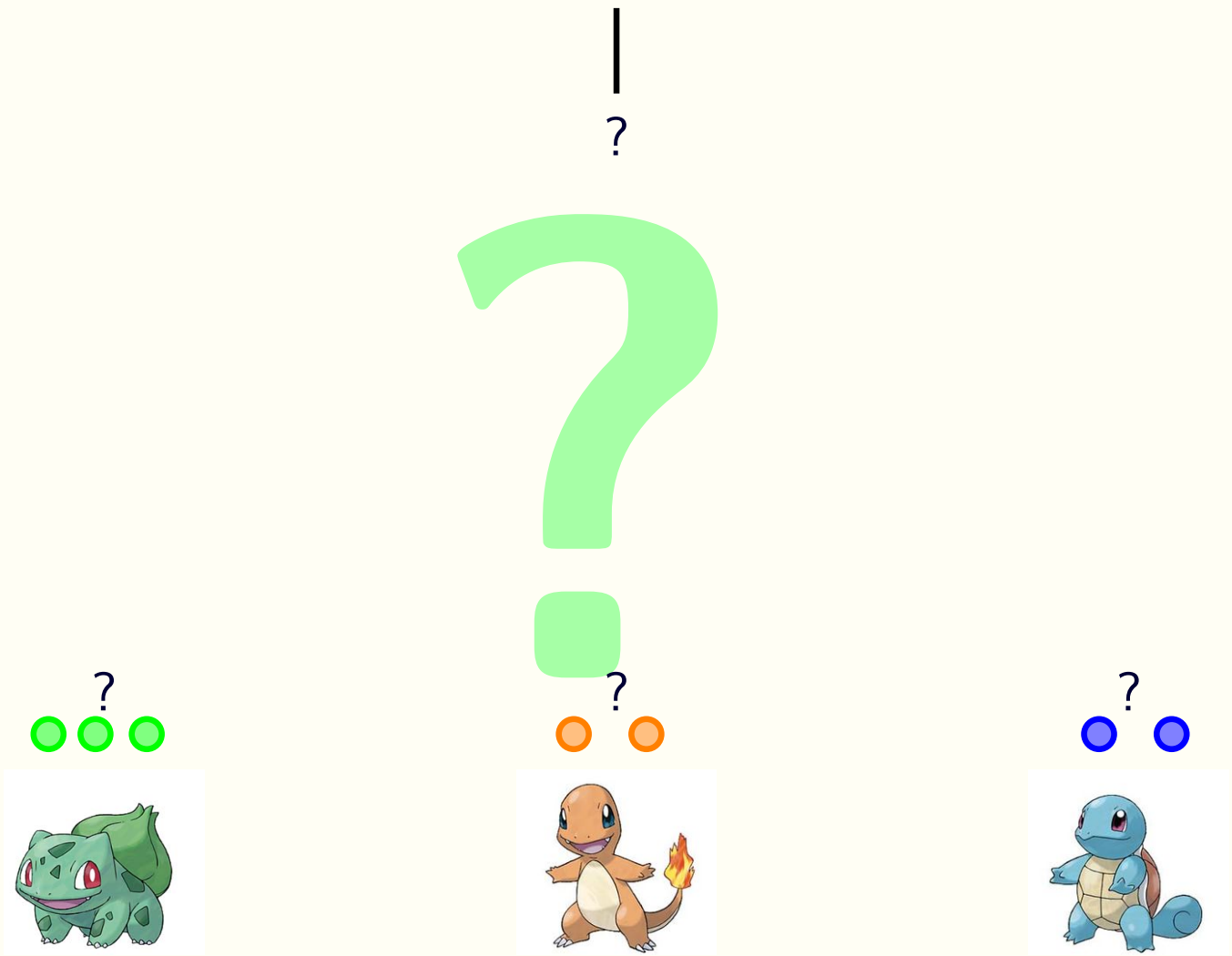
Evolution of Species vs Genes



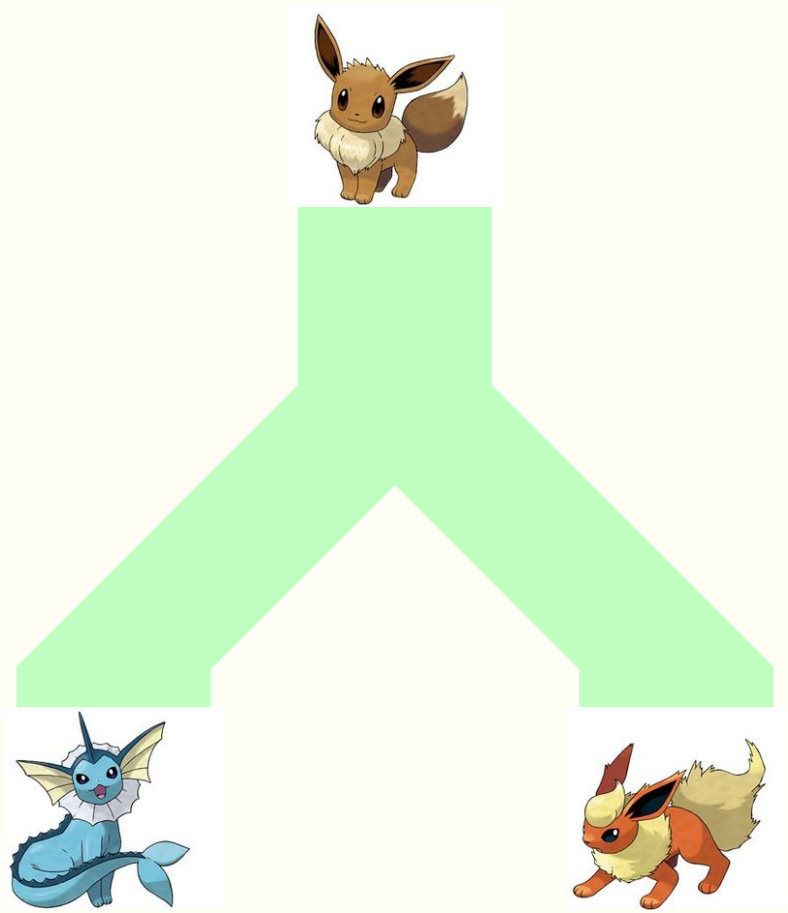
Evolution of Species vs Genes



Evolution of Species vs Genes

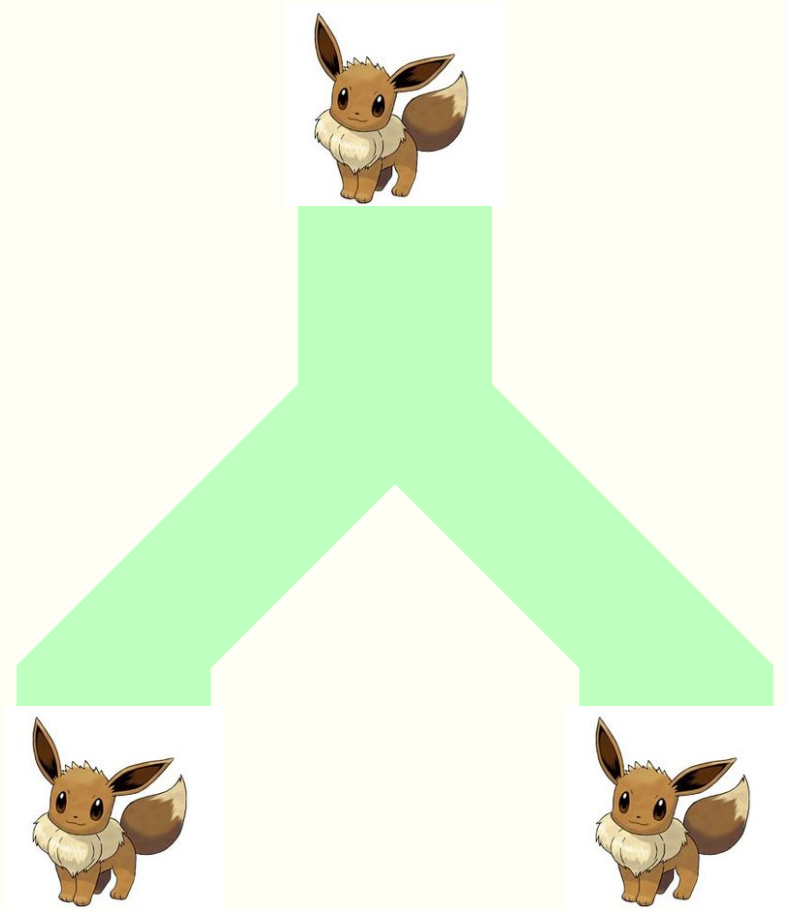


Propagation of genes: Speciation



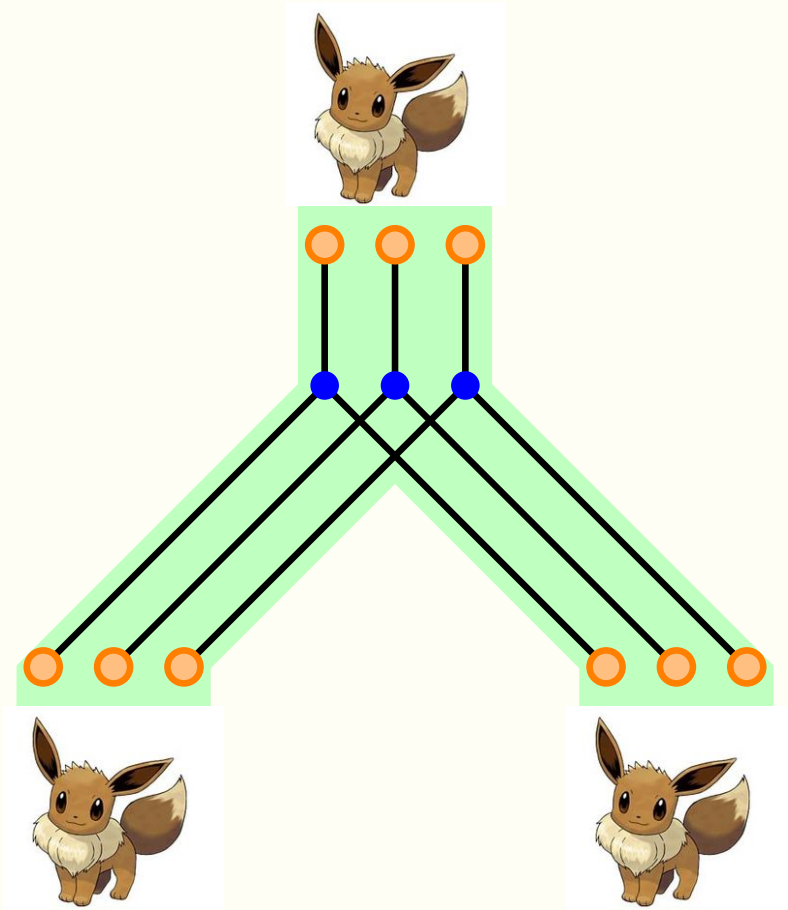
- A **speciation** event occurs when an ancestor species passes a copy of each of its genes onto its descendants.
- A pair of genes are **orthologs** if they are most closely related by a speciation event.

Propagation of genes: Speciation



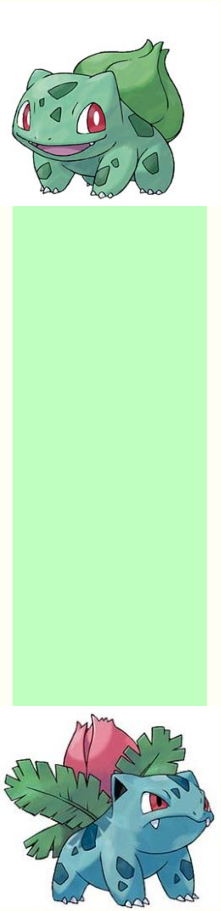
- A **speciation** event occurs when an ancestor species passes a copy of each of its genes onto its descendants.
- A pair of genes are **orthologs** if they are most closely related by a speciation event.

Propagation of genes: Speciation



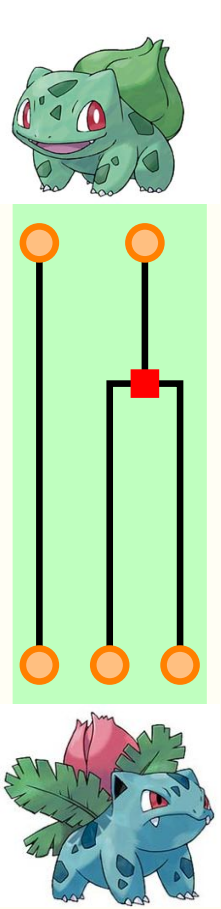
- A **speciation** event occurs when an ancestor species passes a copy of each of its genes onto its descendants.
- A pair of genes are **orthologs** if they are most closely related by a speciation event.

Propagation of genes: Duplication



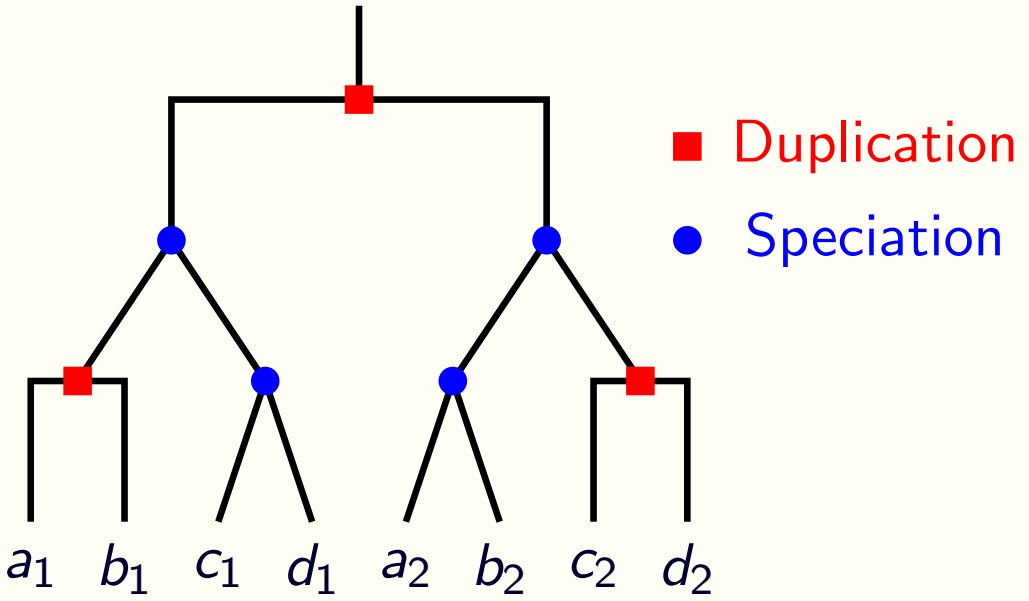
- A **duplication** event occurs when a species gains an additional copy of a gene.
- A pair of genes are **paralogs** if they are most closely related by a duplication event.

Propagation of genes: Duplication



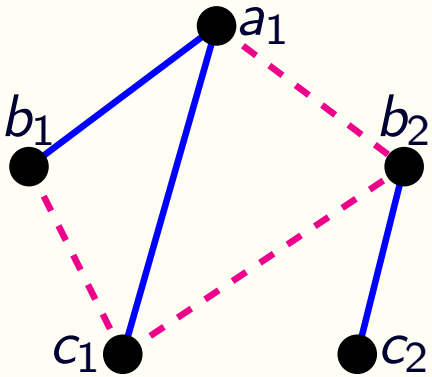
- A **duplication** event occurs when a species gains an additional copy of a gene.
- A pair of genes are **paralogs** if they are most closely related by a duplication event.

Orthologs vs paralogs



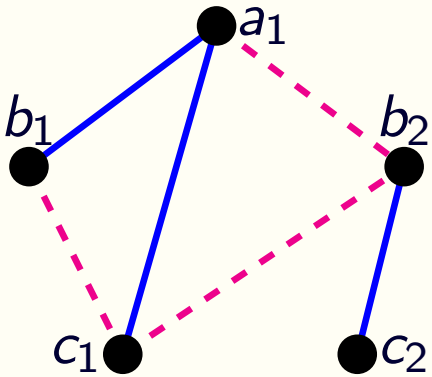
- We depict the evolutionary history of a set of genes with a **DS-tree**.
- Two genes are **orthologs** if they are related by a speciation event, and **paralogs** if they are related by a duplication event.
- Useful for functional analysis - orthologs tend to perform similar functions, paralogs tend to differ.

Constraint Graphs



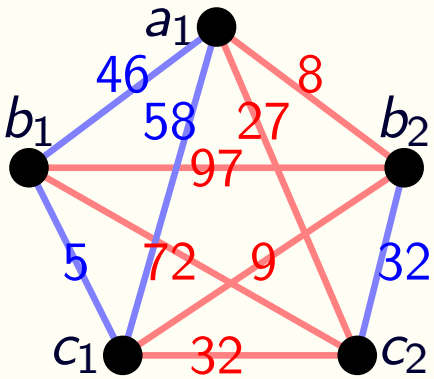
- A **constraint graph** $G = (V, E \cup \mathit{Unknown})$ represents the known orthology / paralogy relations between pairs of genes.
- Solid edges = known orthologs; non-edges = known paralogs; dashed edges = unknown.

Constraint Graphs

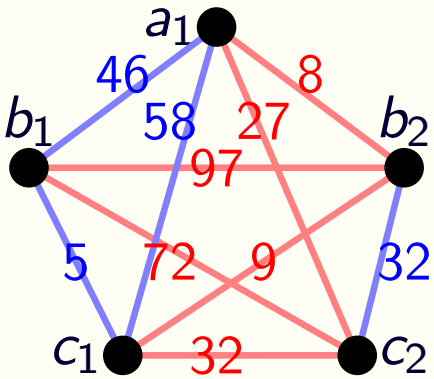


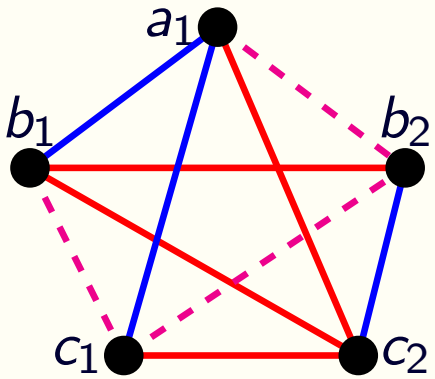
- A **constraint graph** $G = (V, E \cup \mathit{Unknown})$ represents the known orthology / paralogy relations between pairs of genes.
- Solid edges = known orthologs; non-edges = known paralogs; dashed edges = unknown.

Constraint Graphs

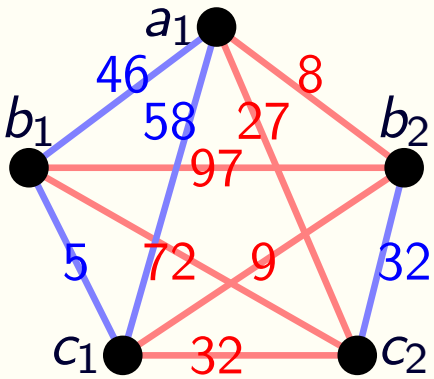


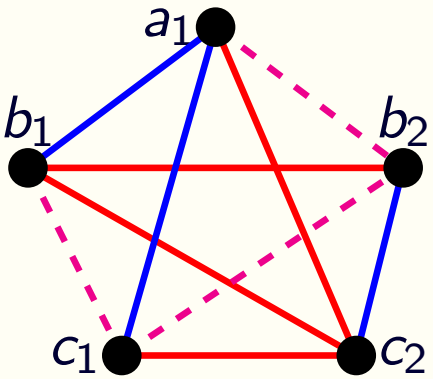
Constraint Graphs

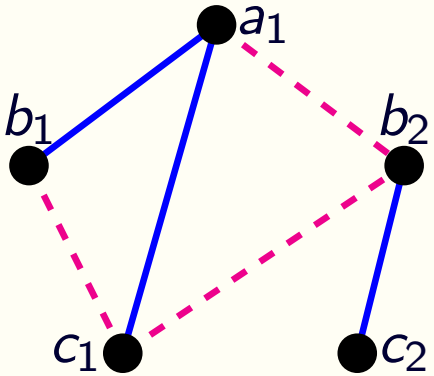




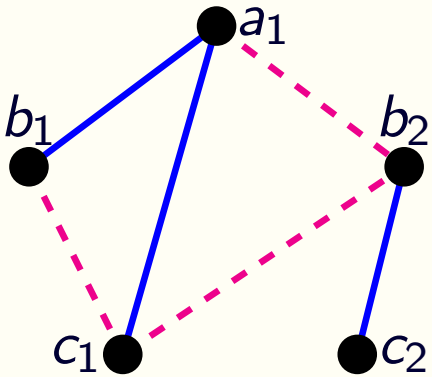
Constraint Graphs

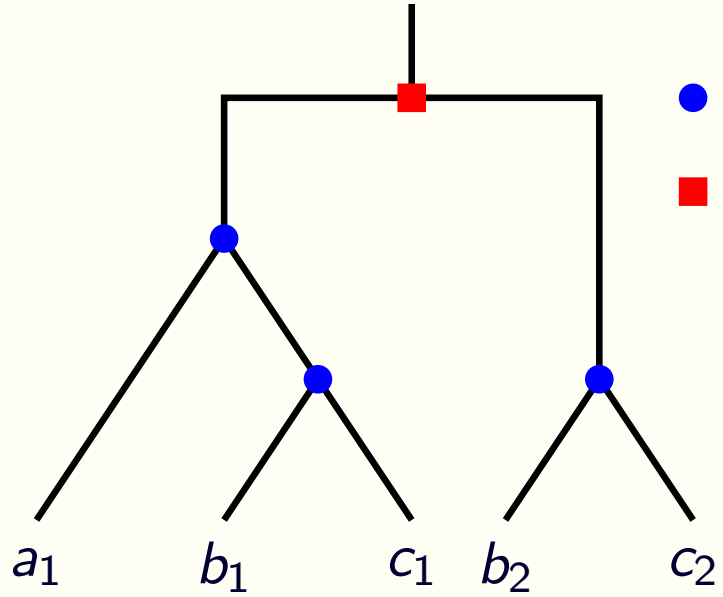






Constraint graphs

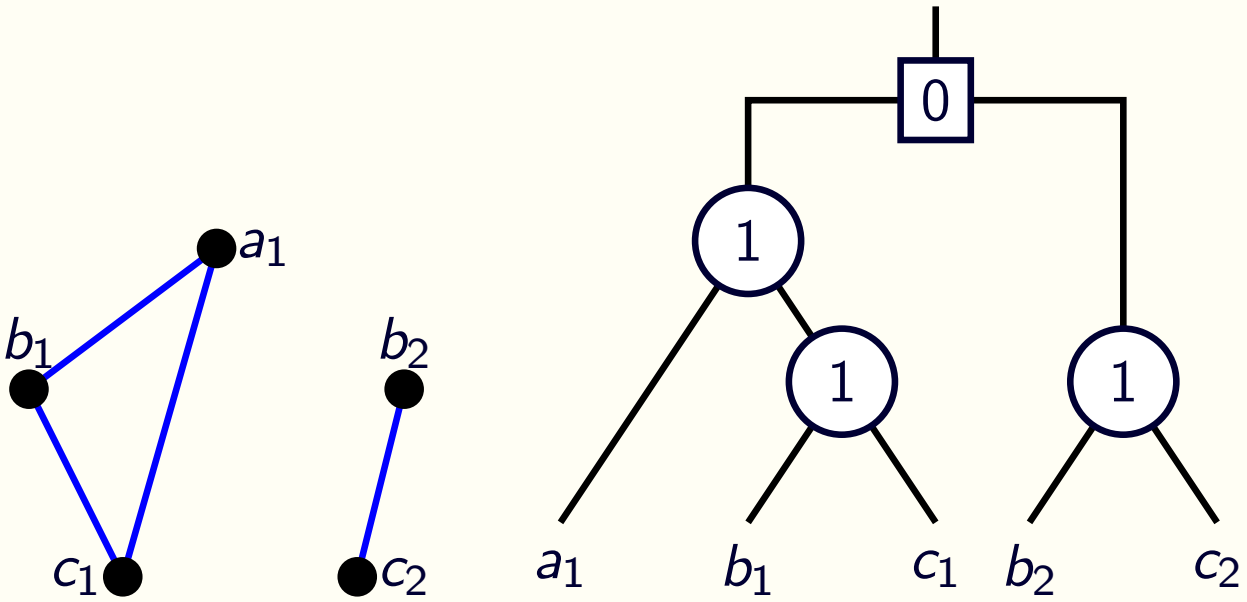




- Speciation
- Duplication

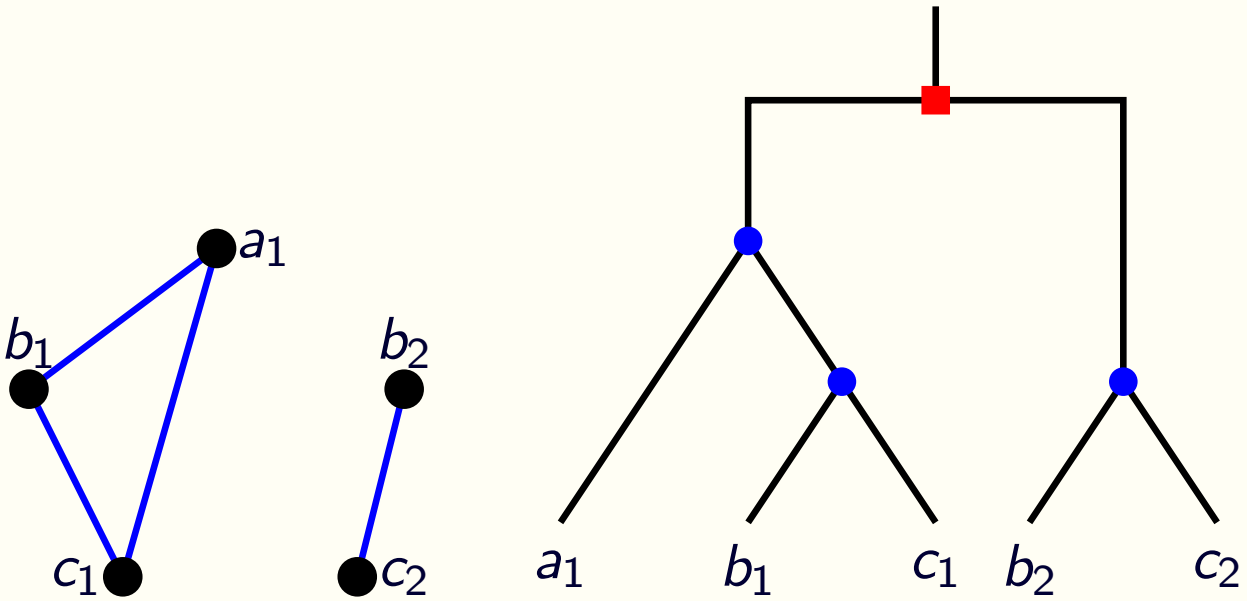
- Given a constraint graph, we wish to find an evolutionary history for the set of genes.
- A **DS-tree** (Duplication/Speciation tree) is a rooted tree over a set of genes, with internal nodes labelled as **duplication** events (red squares) or **speciation** events (blue circles).
- A DS-tree T **satisfies** a constraint graph C if for every pair of known orthologs are related by a speciation event, and every pair of known paralogs are related by a duplication event.

Relation to cographs



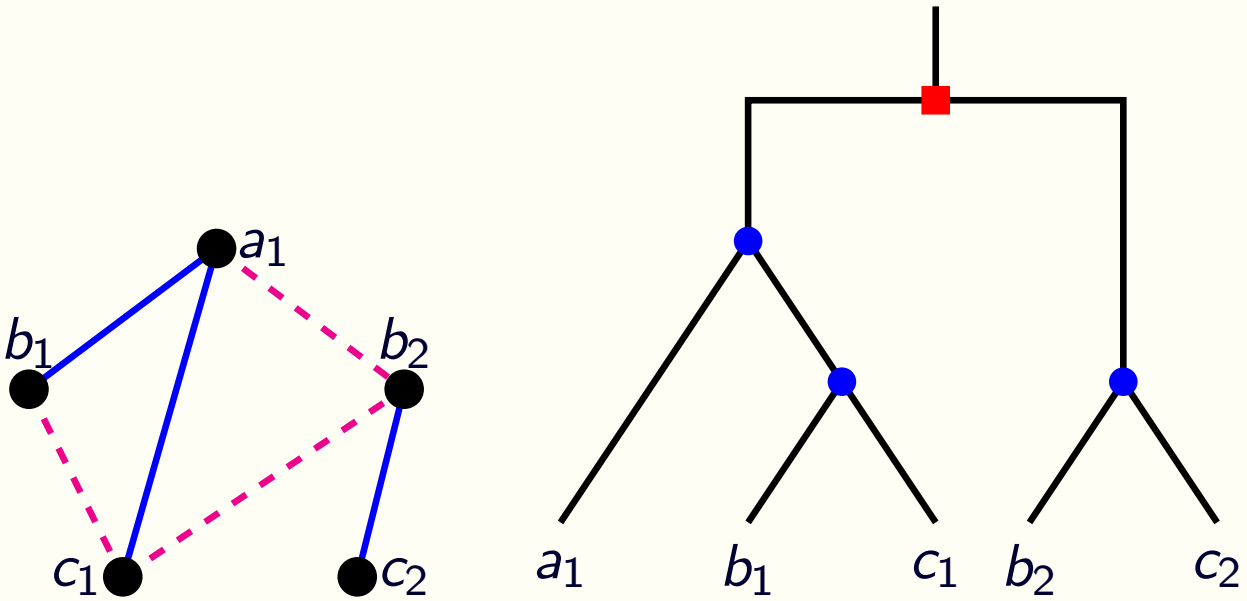
- A graph G with a vertex set V is a **cograph** if there exists a **co-tree** T with leaf set V, internal nodes labelled 0 or 1, such that two vertices are adjacent in G iff their l.c.a. in T with label 1.

Relation to cographs



- A graph G with a vertex set V is a **cograph** if there exists a **co-tree** T with leaf set V, internal nodes labelled 0 or 1, such that two vertices are adjacent in G iff their l.c.a. in T with label 1.

Relation to cographs



- Co-trees are equivalent to (binary) co-trees.
- Finding a DS-tree satisfying a constraint graph is equivalent to the Cograph Sandwich Problem (polynomial-time solveable[1]).

[1] Golumbic, M., Kaplan, H., Shamir, R.: Graph sandwich problems. Journal of Algorithms (1995)

Finding a DS-tree for a cograph

Cotree($G = (V, E)$):

1 **If** G is disconnected:
2 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \cap E = \emptyset$;
3 Set $root.type = Dup$;
4 $root.add\_child(\text{Cotree}(G[V_1]))$;
5 $root.add\_child(\text{Cotree}(G[V_2]))$;
6 **Return** $root$;
7 **Else**:
8 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \subseteq E$;
9 Set $root.type = Spec$;
10 $root.add\_child(\text{Cotree}(G[V_1]))$;
11 $root.add\_child(\text{Cotree}(G[V_2]))$;
12 **Return** $root$;

Finding a DS-tree for a cograph

Cotree($G = (V, E)$):

1 **If** G is disconnected:
2 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \cap E = \emptyset$;
3 Set $root.type = Dup$;
4 $root.add\_child(\text{Cotree}(G[V_1]))$;
5 $root.add\_child(\text{Cotree}(G[V_2]))$;
6 **Return** $root$;
7 **Else**:
8 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \subseteq E$;
Exception: If no partition found, G is not a cograph
9 Set $root.type = Spec$;
10 $root.add\_child(\text{Cotree}(G[V_1]))$;
11 $root.add\_child(\text{Cotree}(G[V_2]))$;
12 **Return** $root$;

Checking if a constraint graph has a DS-tree

SandwichCograph($G = (V, E \cup \mathit{Unknown})$):

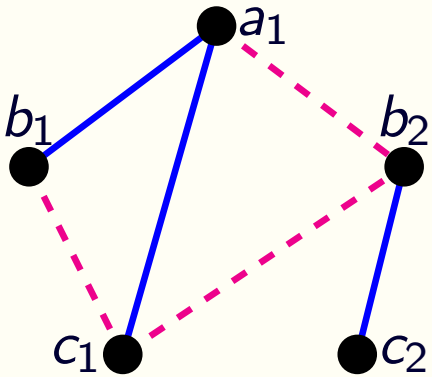
1 **If** $G[E]$ is disconnected:
2 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \cap E = \emptyset$;
3 Set $\mathit{root}.\mathit{type} = \mathit{Dup}$;
4 **Else**:
5 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \subseteq E \cup \mathit{Unknown}$;
6 Set $\mathit{root}.\mathit{type} = \mathit{Spec}$;
7 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_1]$));
8 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_2]$));
9 **Return** root;

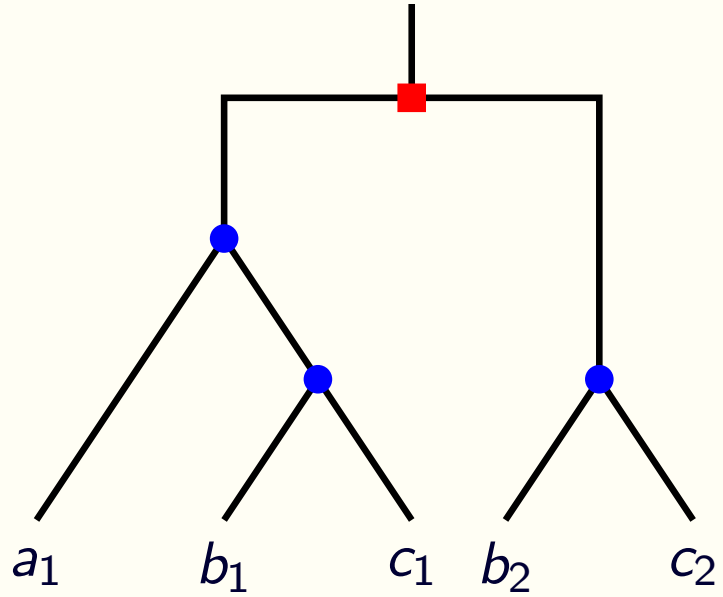
Checking if a constraint graph has a DS-tree

SandwichCograph($G = (V, E \cup \mathit{Unknown})$):

1 **If** $G[E]$ is disconnected:
2 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \cap E = \emptyset$;
3 Set $\mathit{root}.\mathit{type} = \mathit{Dup}$;
4 **Else**:
5 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \subseteq E \cup \mathit{Unknown}$;
Exception: If no partition found, G has no DS-tree
6 Set $\mathit{root}.\mathit{type} = \mathit{Spec}$;
7 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_1]$));
8 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_2]$));
9 **Return** root;

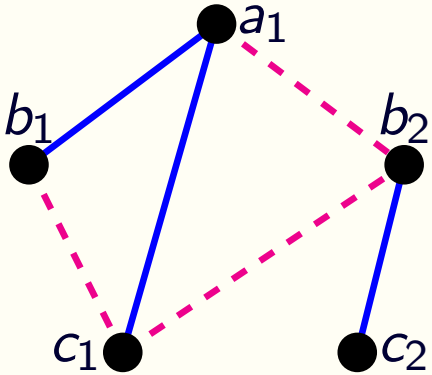
Finding a DS-tree

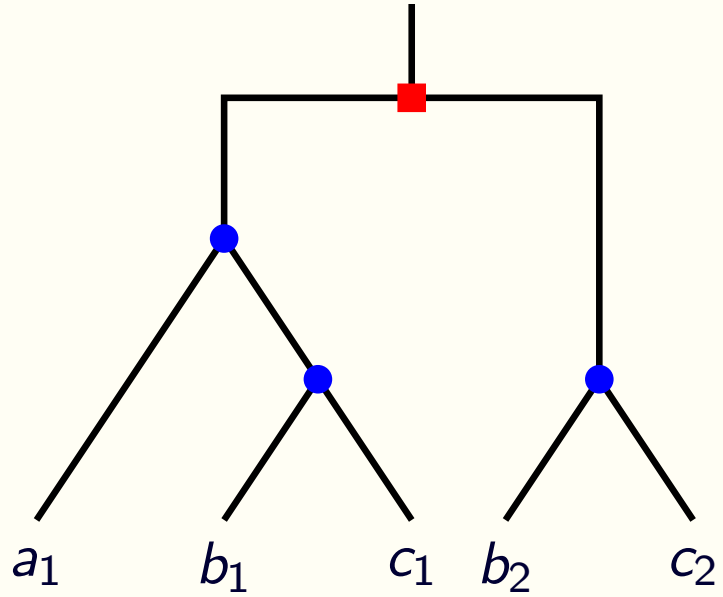




- A DS-tree T **satisfies** a constraint graph $G = (V, E \cup \mathit{Unknown})$ if for every pair of known orthologs are related by a speciation event, and every pair of known paralogs are related by a duplication event.
- Finding a DS-tree satisfying a constraint graph can be done in polynomial time.
- HOWEVER...
- ... not every DS-tree satisfying G is a good solution.

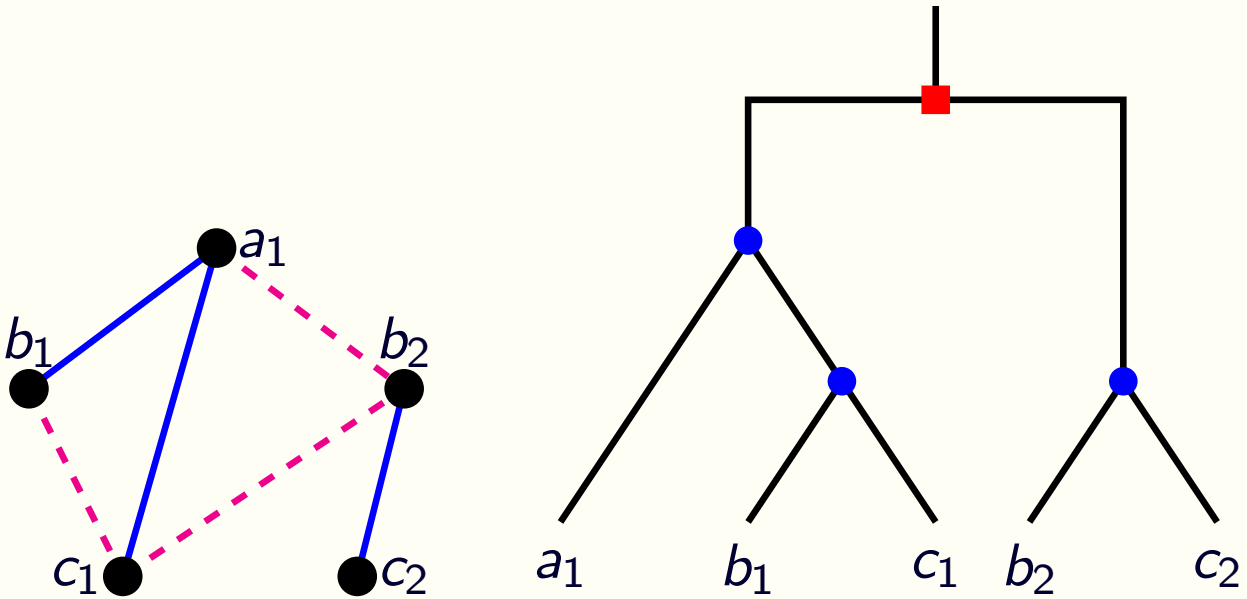
Finding a DS-tree





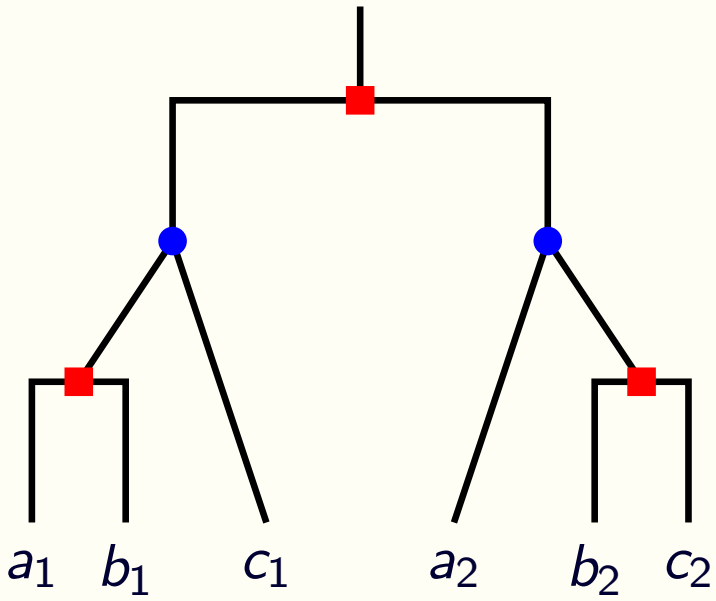
- A DS-tree T **satisfies** a constraint graph $G = (V, E \cup \mathit{Unknown})$ if for every pair of known orthologs are related by a speciation event, and every pair of known paralogs are related by a duplication event.
- Finding a DS-tree satisfying a constraint graph can be done in polynomial time.
- HOWEVER...
- ... not every DS-tree satisfying G is a good solution.

Finding a DS-tree



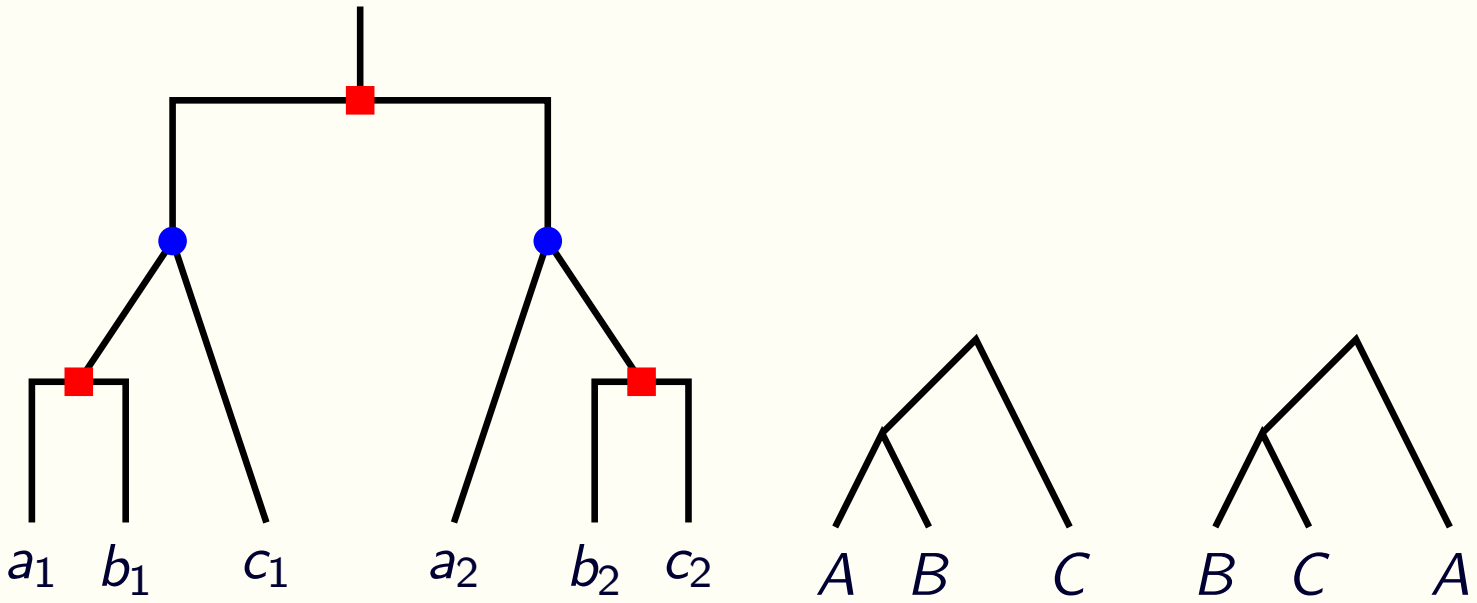
- A DS-tree T **satisfies** a constraint graph $G = (V, E \cup \mathit{Unknown})$ if for every pair of known orthologs are related by a speciation event, and every pair of known paralogs are related by a duplication event.
- Finding a DS-tree satisfying a constraint graph can be done in polynomial time.
- HOWEVER...
- ... not every DS-tree satisfying G is a good solution.

When DS-trees go wrong



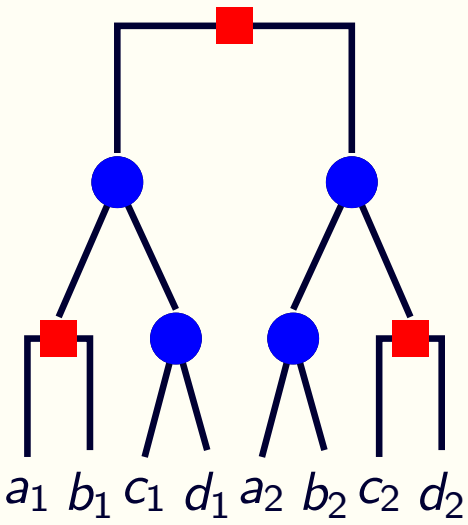
- Some DS-trees can provide contradictory evidence about the structure of the species tree.

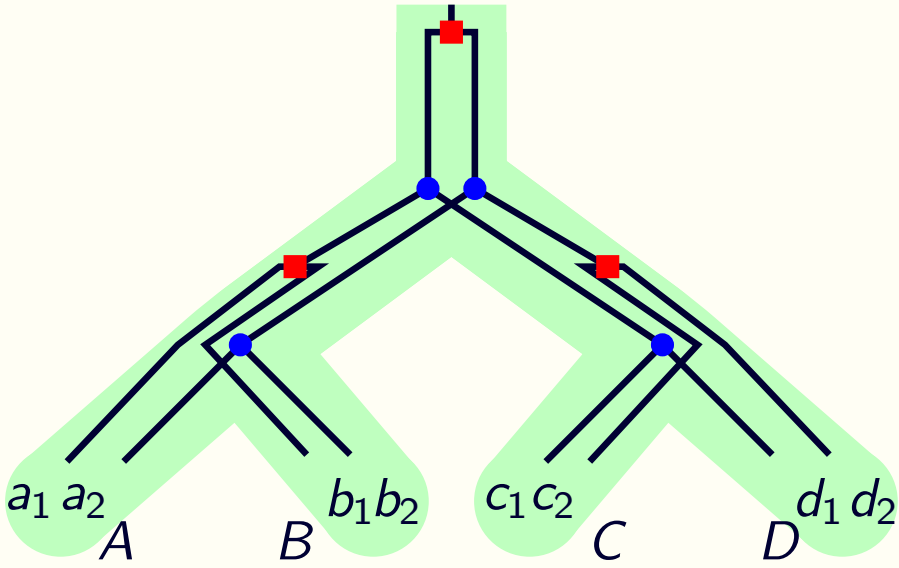
When DS-trees go wrong



- Some DS-trees can provide contradictory evidence about the structure of the species tree.

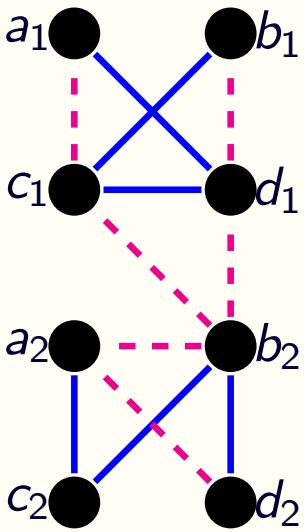
Consistency with a species tree

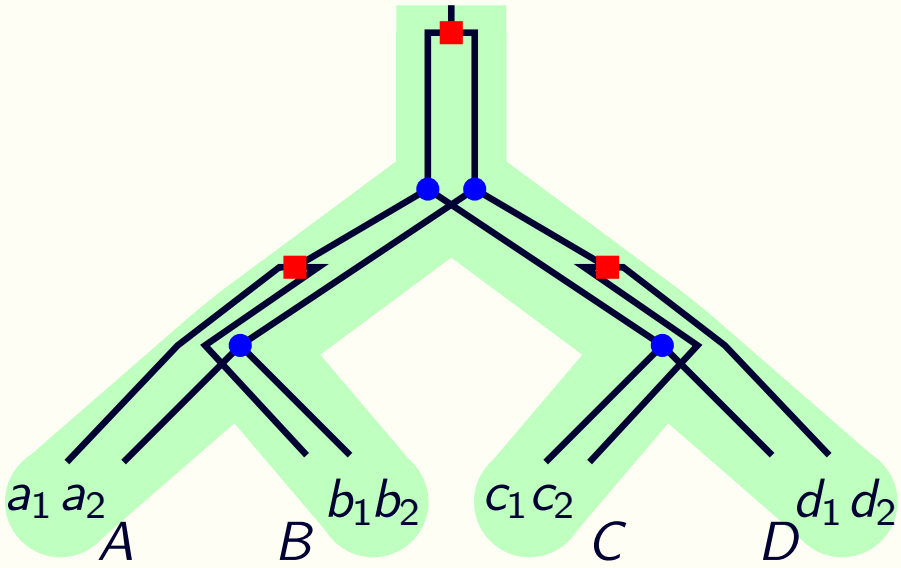




- DS-tree T is **consistent** with species tree S if T can be embedded in S s.t. every speciation coincides with a branching in S, and every gene is mapped to its species.
- More formal (equivalent) definition:
  Given a DS-tree T on V and species assignment $s : V \to \Sigma$, and a species tree S on Σ, T is **consistent** with S if for every speciation event z in T, and distinct children x, y of z, $lca_S(s(leaf_T(x)))$ and $lca_S(s(leaf_T(y)))$ are separated in S.

Problem Definition





- **Input:** A constraint graph $G = (V, E \cup \mathit{Unknown})$ on vertex set V; species mapping $s : V \to \Sigma$
- **Output:** A DS-tree T on V; species tree S on Σ; s.t. T *satisfies* G and is *consistent* with S.

Previous Results

Input: A constraint graph $G = (V, E \cup \mathit{Unknown})$ on vertex set V; species mapping $s : V \to \Sigma$
Output: A DS-tree T on V; species tree S on Σ; s.t. T *satisfies* G and is *consistent* with S.

- Polynomial time algorithm when S is known (Lafond & El-Mabrouk, 2014[2])
- Polynomial tile algorithm when T is known i.e. G has no unknown pairs (using BUILD algorithm[3]).
- **Our result:** Polynomial time algorithm for general case (S and T unknown).

[2] Lafond, M., El-Mabrouk, N.: Orthology and paralogy constraints: satisfiability and consistency. BMC genomics (2014)

[3] Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. (1981)

Previous Results

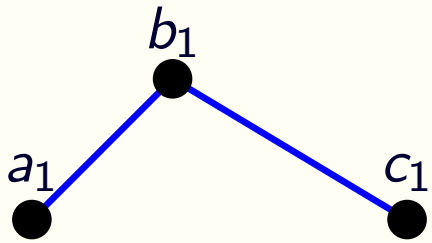
Input: A constraint graph $G = (V, E \cup \mathit{Unknown})$ on vertex set V; species mapping $s : V \to \Sigma$
Output: A DS-tree T on V; species tree S on Σ; s.t. T *satisfies* G and is *consistent* with S.

- Polynomial time algorithm when S is known (Lafond & El-Mabrouk, 2014[2])
- Polynomial tile algorithm when T is known i.e. G has no unknown pairs (using BUILD algorithm[3]).
- **Our result:** Polynomial time algorithm for general case (S and T unknown).

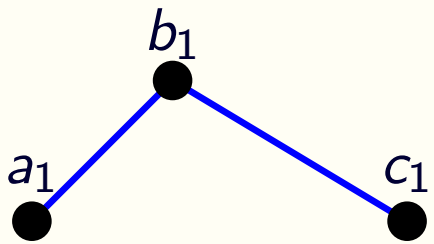
[2]Lafond, M., El-Mabrouk, N.: Orthology and paralogy constraints: satisfiability and consistency. BMC genomics (2014)

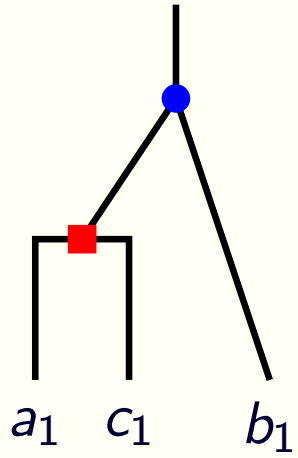
[3]Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. (1981)

Simple example: three genes

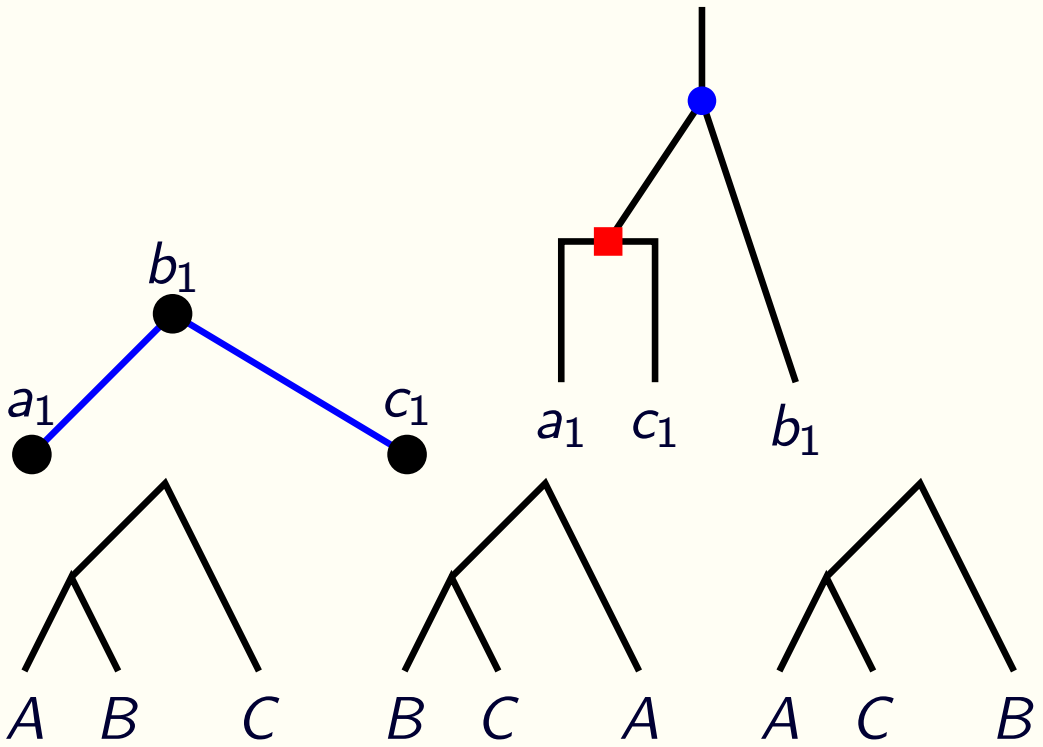


Simple example: three genes

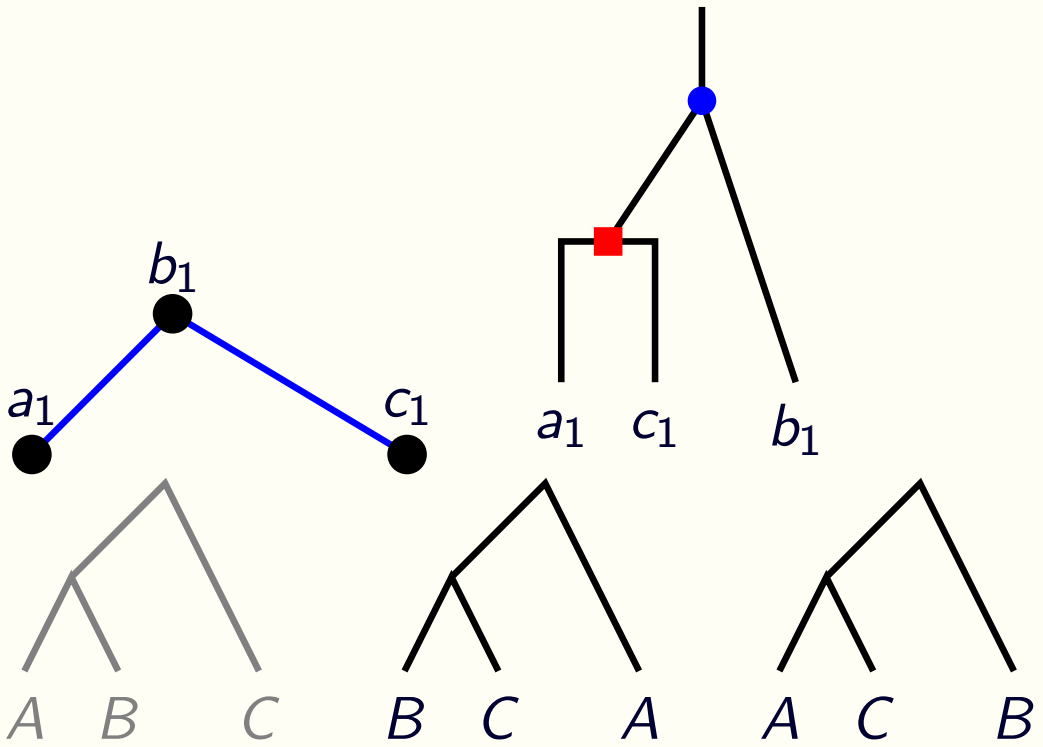




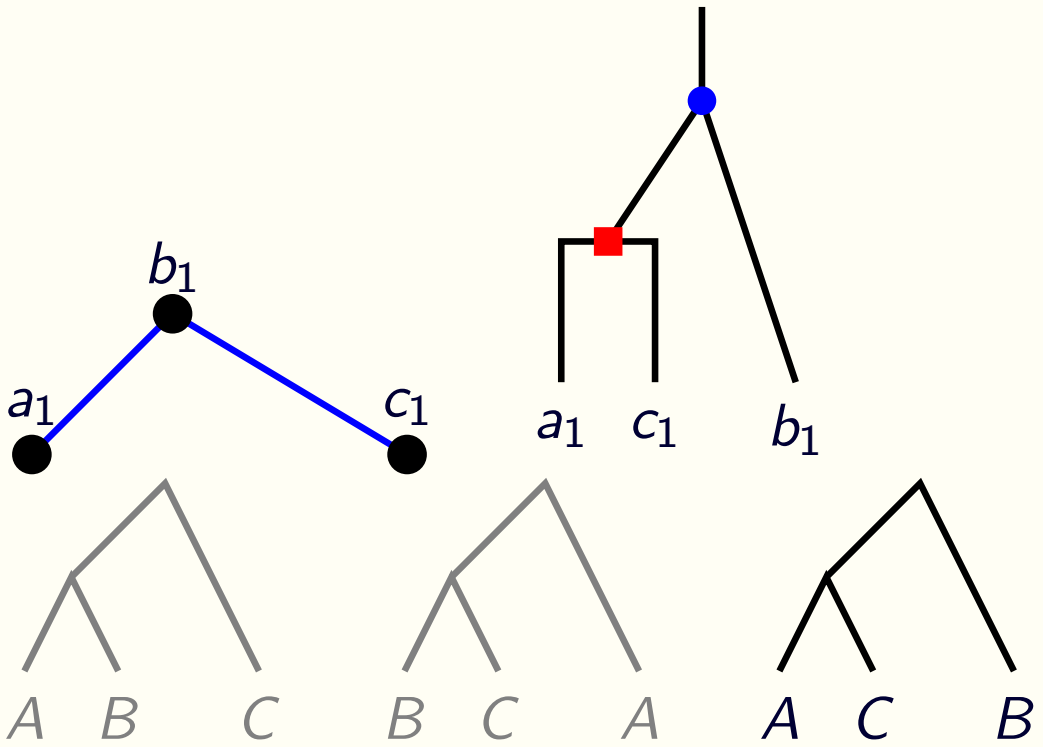
Simple example: three genes



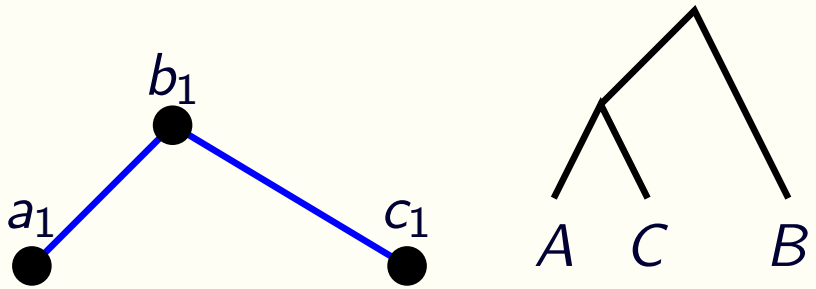
Simple example: three genes



Simple example: three genes



Simple example: three genes

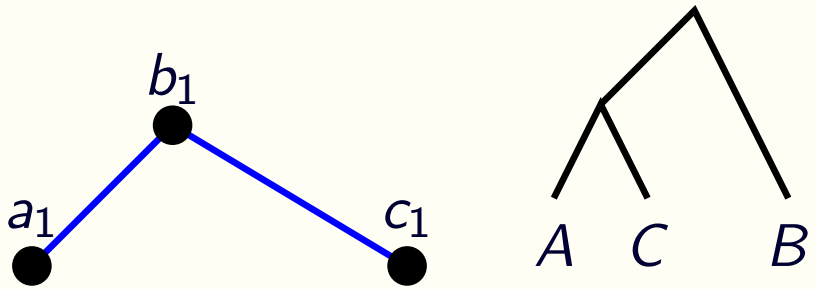


- Given a graph $G = (V, E)$ (no unknown pairs), define $P_3(G) = \{s(x)s(y)|s(z) : xy \in E, yz \in E, xz \notin E\}$.
- For any $AC|B \in P_3(G)$, we must have that $AC|B$ is displayed by the species tree.

Theorem (Lafond & El-Mabrouk, 2014)

Let $G = (V, E)$ have a DS-tree T and let $s : V \to \Sigma$ be a species assignment. Let S be a species tree on Σ. Then T is consistent with S if and only if S displays all triplets in $P_3(G)$.

Simple example: three genes



- Given a graph $G = (V, E)$ (no unknown pairs), define $P_3(G) = \{s(x)s(y)|s(z) : xy \in E, yz \in E, xz \notin E\}$.
- For any $AC|B \in P_3(G)$, we must have that $AC|B$ is displayed by the species tree.

Theorem (Lafond & El-Mabrouk, 2014)

Let $G = (V, E)$ have a DS-tree T and let $s : V \to \Sigma$ be a species assignment. Let S be a species tree on Σ. Then T is consistent with S if and only if S displays all triplets in $P_3(G)$.

Consistency dependant on triplets

Theorem (Lafond & El-Mabrouk, 2014)

Let $G = (V, E)$ have a DS-tree T and let $s : V \to \Sigma$ be a species assignment. Let S be a species tree on Σ. Then T is consistent with S if and only if S displays all triplets in $P_3(G)$.

- Thus if T is known, we just need an algorithm to determine consistency of a set of triples.
- Such an algorithm exists (BUILD algorithm[4]).

[4] Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. (1981)

Consistency dependant on triplets

Theorem (Lafond & El-Mabrouk, 2014)

Let $G = (V, E)$ have a DS-tree T and let $s : V \to \Sigma$ be a species assignment. Let S be a species tree on Σ. Then T is consistent with S if and only if S displays all triplets in $P_3(G)$.

- Thus if T is known, we just need an algorithm to determine consistency of a set of triples.
- Such an algorithm exists (BUILD algorithm[4]).

[4] Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. (1981)

Algorithm when T is known

SpeciesTree($G = (V, E), s : V \to \Sigma$):

1 Set $H = (\Sigma, F = \emptyset)$;
2 **For**$(AC|B \in P_3(G))$:
3 Set $F = F \cup \{AC\}$;
4 **If** H is disconnected:
5 Find partition $\Sigma = X \uplus Y$ such that $X \times Y \cap F = \emptyset$;
6 $root.add\_child$(SpeciesTree($G[s^{-1}(X)], s_X$);
7 $root.add\_child$(SpeciesTree($G[s^{-1}(Y)], s_Y$);
8 **Return** $root$;
9 **Else**:
10 **Return** NULL;

Algorithm when S is known

ConsistentSandwichCograph($G = (V, E \cup \mathit{Unknown}), S, s : V \to V(S)$):

1 **If** $G - \mathit{Unknown}$ is disconnected:
2 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \cap E = \emptyset$;
3 Set $\mathit{root}.\mathit{type} = \mathit{Dup}$;
4 **Else**:
5 Find partition $V = V_1 \uplus V_2$ such that $V_1 \times V_2 \subset E \cup \mathit{Unknown}$ and $s(V_1), s(V_2)$ are separated in S;
6 Set $\mathit{root}.\mathit{type} = \mathit{Spec}$;
7 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_1]$));
8 $\mathit{root}.\mathit{add\_child}$(SandwhichCograph($G[V_2]$));
9 **Return** root;

Our main result

Theorem

In $O(n^3)$ time, given a constraint graph C on n genes, we can find a DS-tree T satisfying C, and a species S consistent with T (if such S, T exist).

- Proof combines previous algorithms for cases when S known/ T known.
- Can be used to provide evidence for sturcture of a species tree in cases where this is uncertain.

Our main result

Theorem

In $O(n^3)$ time, given a constraint graph C on n genes, we can find a DS-tree T satisfying C, and a species S consistent with T (if such S, T exist).

- Proof combines previous algorithms for cases when S known/ T known.
- Can be used to provide evidence for sturcture of a species tree in cases where this is uncertain.

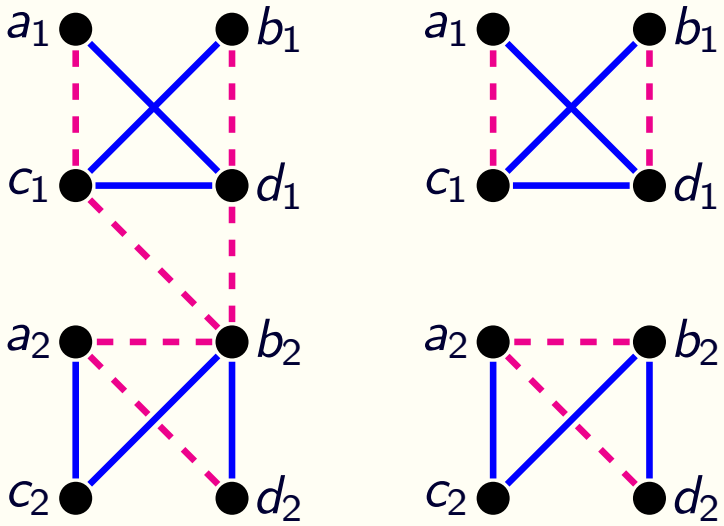
Our main result

Theorem

In $O(n^3)$ time, given a constraint graph C on n genes, we can find a DS-tree T satisfying C, and a species S consistent with T (if such S, T exist).

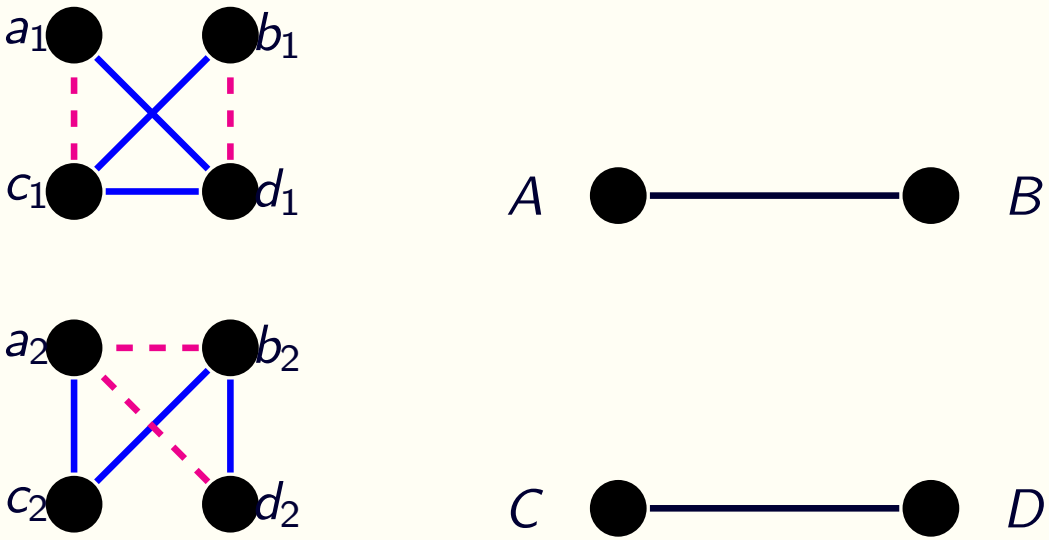
- Proof combines previous algorithms for cases when S known/ T known.
- Can be used to provide evidence for sturcture of a species tree in cases where this is uncertain.

Algorithm Sketch



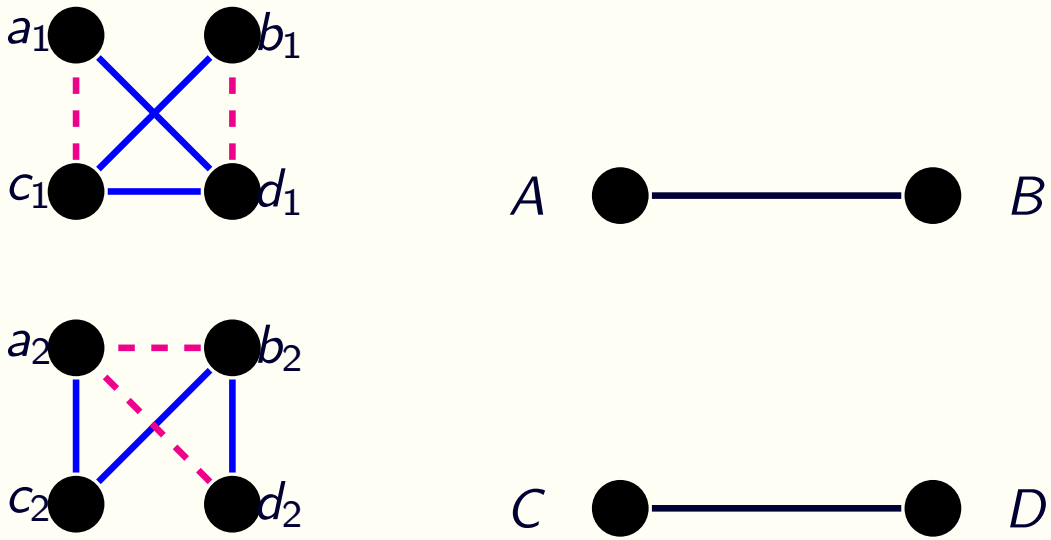
- Mark as paralog every unknown edge xy such that x and y belong to distinct connected components of known orthologs.

Algorithm Sketch



- Construct a species graph H by connecting A, B iff there exist gene x in species A, y in species B, such that x, y are in the same component of C and xy are known paralogs.
- If H is connected, there is no solution.
- If H can be partitoned into disjoint parts $\mathcal{X}, \mathcal{Y}$, we can assume the first branch of our species tree separates $\mathcal{X}$ and $\mathcal{Y}$

Algorithm Sketch

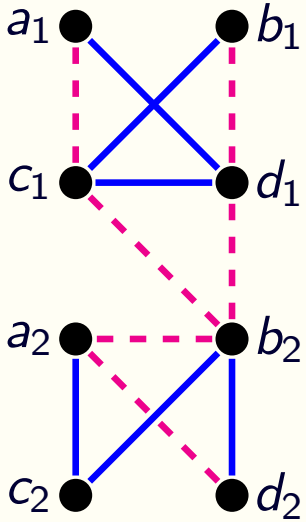


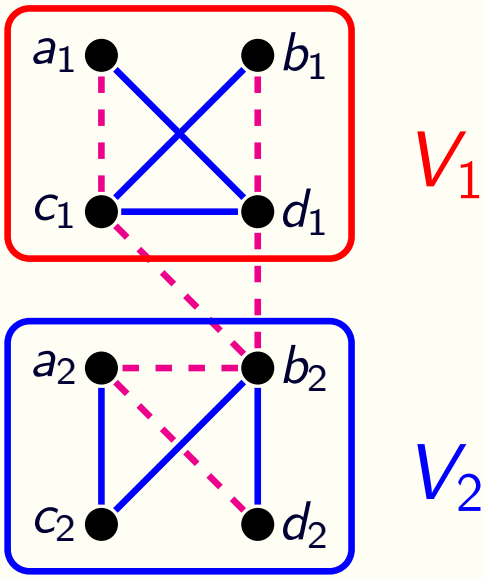
- Construct a species graph H by connecting A, B iff there exist gene x in species A, y in species B, such that x, y are in the same component of C and xy are known paralogs.
- If H is connected, there is no solution.
- If H can be partitoned into disjoint parts $\mathcal{X}, \mathcal{Y}$, we can assume the first branch of our species tree separates $\mathcal{X}$ and $\mathcal{Y}$

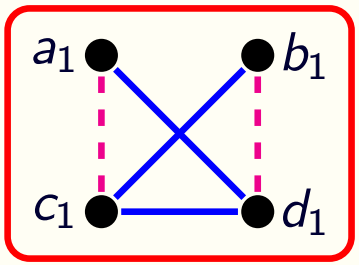
Our Algorithm

ConsistentSandwichCograph($G = (V, E \cup \mathit{Unknown}), s : V \to \Sigma$):

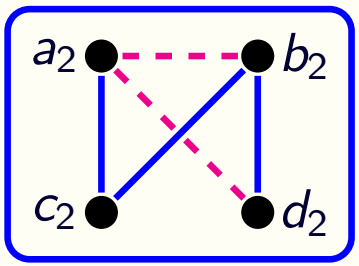
1 **If** $G - \mathit{Unknown}$ is disconnected:
2 Delete edges in *Unknown* between components of $G[E]$;
3 **For**$(x, z \in V)$:
4 **If**$(x, z$ in same componet of $G[E]$ but $xy \notin E \cup \mathit{Unknown})$:
5 Add $s(x)s(z)$ as an edge in H;
6 **If** $H = (F, \Sigma)$ is disconnected:
7 Find partition $\Sigma = X \uplus Y$ such that $X \times Y \cap F = \emptyset$;
8 $\mathit{root}.\mathit{add\_child}(\text{SpeciesTree}(G[s^{-1}(X)], s_X)$;
9 $\mathit{root}.\mathit{add\_child}(\text{SpeciesTree}(G[s^{-1}(Y)], s_Y)$;
10 **Return** *root*;


a_1
b_1
c_1
d_1
a_2
b_2
c_2
d_2

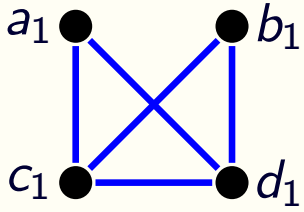

a_1
b_1
c_1
d_1
V_1
a_2
b_2
c_2
d_2
V_2

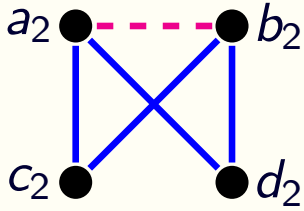


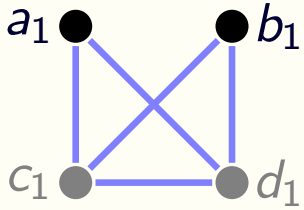
V_1

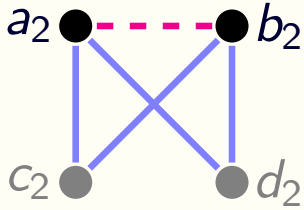


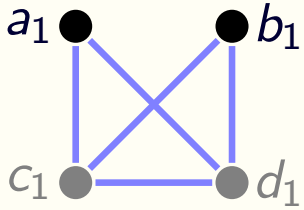
V_2

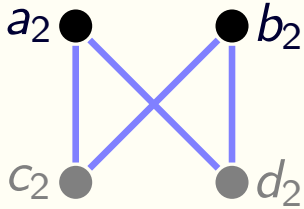

a_1
b_1
c_1
d_1

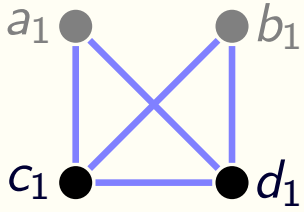

a_2
b_2
c_2
d_2

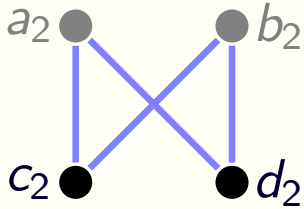

a_1
b_1
c_1
d_1

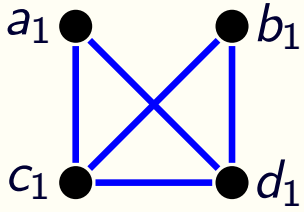

a_2
b_2
c_2
d_2

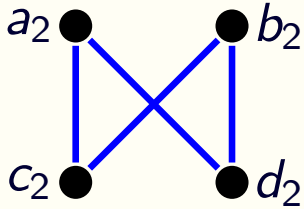

a_1
b_1
c_1
d_1


a_2
b_2
c_2
d_2

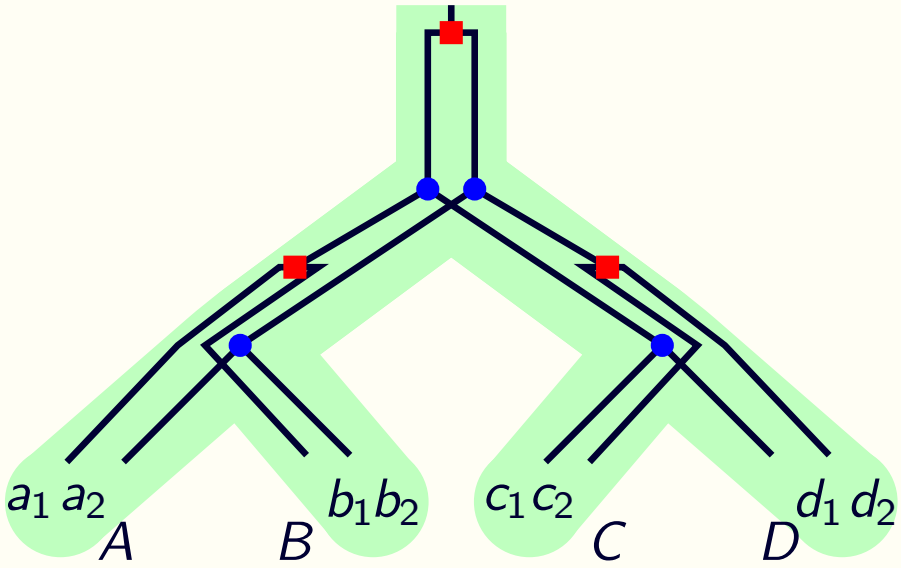

a_1
b_1
c_1
d_1


a_2
b_2
c_2
d_2


a_1
b_1
c_1
d_1


a_2
b_2
c_2
d_2

Algorithm Sketch



- Construct a species graph H by connecting A, B iff there exist gene x in species A, y in species B, such that x, y are in the same component of C and xy are known paralogs.
- If H is connected, there is no solution.
- If H can be partitoned into disjoint parts $\mathcal{X}, \mathcal{Y}$, we can assume the first branch of our species tree separates $\mathcal{X}$ and $\mathcal{Y}$

Minimizing Duplications

Input: A constraint graph C on vertex set V; species mapping $s : V \to \Sigma$

Output: A DS-tree T on V; species tree S on Σ; s.t. T *satisfies* C and is *consistent* with S and T has *minium* number of duplication events.

- This problem is NP-hard (reduction from graph coloring).
- NP-hard even to decide if 2 duplications is possible.
- For any $\epsilon > 0$, NP-hard to decide within a factor of $n^{1-\epsilon}$.

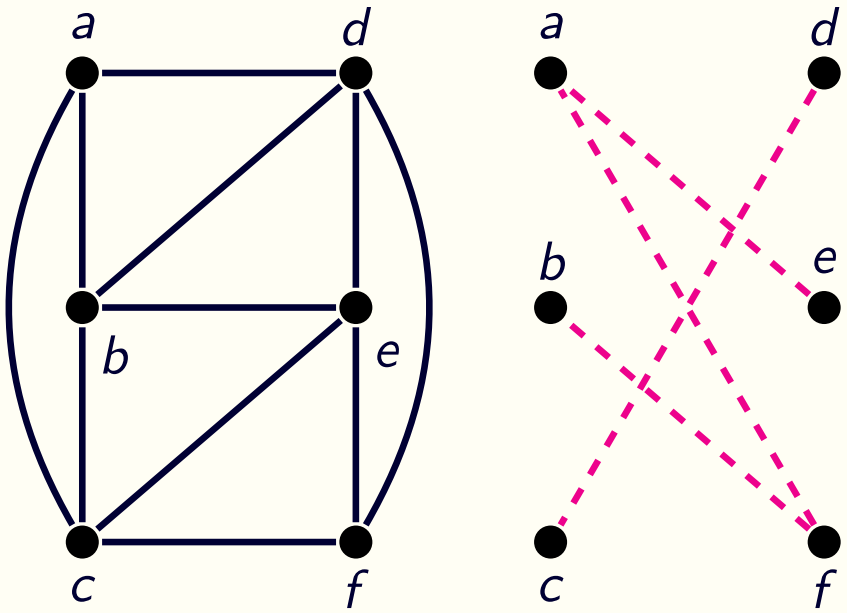
Minimizing Duplications

Input: A constraint graph C on vertex set V; species mapping $s : V \to \Sigma$

Output: A DS-tree T on V; species tree S on Σ; s.t. T *satisfies* C and is *consistent* with S and T has *minium* number of duplication events.

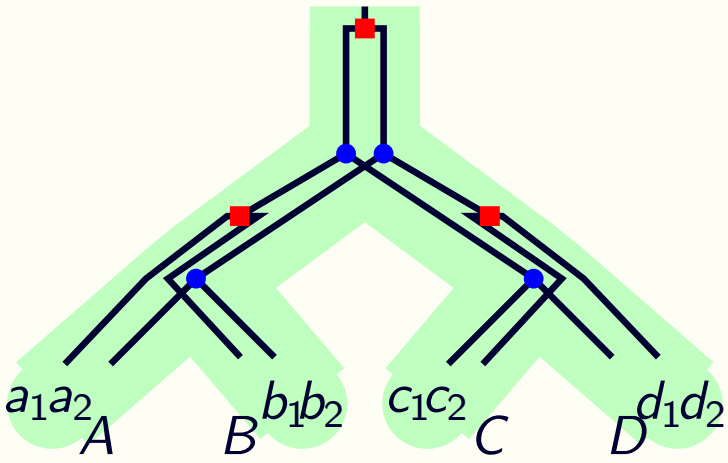
- This problem is NP-hard (reduction from graph coloring).
- NP-hard even to decide if 2 duplications is possible.
- For any $\epsilon > 0$, NP-hard to decide within a factor of $n^{1-\epsilon}$.

Reduction from graph coloring



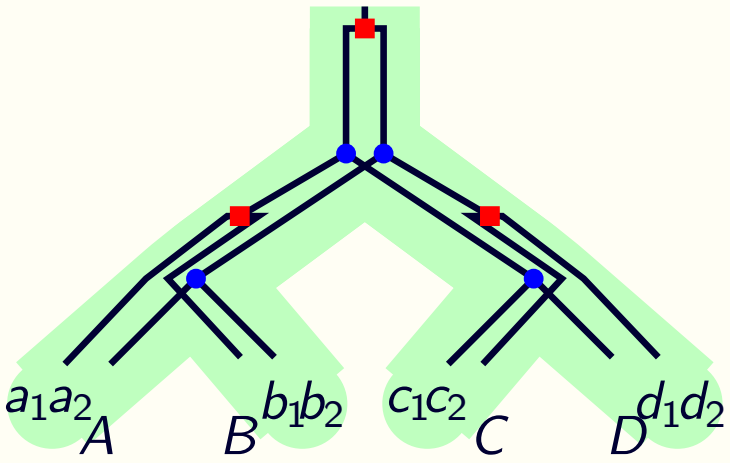
- Given a graph $H = (V, F)$, let constraint graph $G = (V, E \cup \mathit{Unknown})$, $E = \emptyset$, $\mathit{Unknown} = \bar{F}$. Then H is $k+1$-colorable iff G has a DS-tree T with at most k duplications.

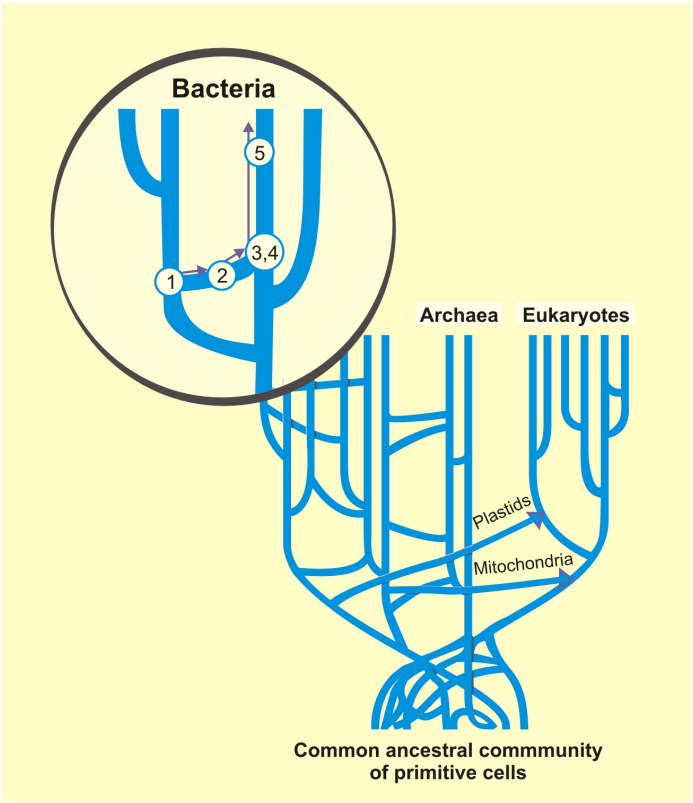
Extending to transfer networks



- We have been assuming that all gene propagation occurs in a species **tree**.
- In practice, **lateral gene transfer** can occur between branches.
- Open questions: Can we solve satisfy constraint graphs if the DS-tree must be "embedded" in an **acyclic network**

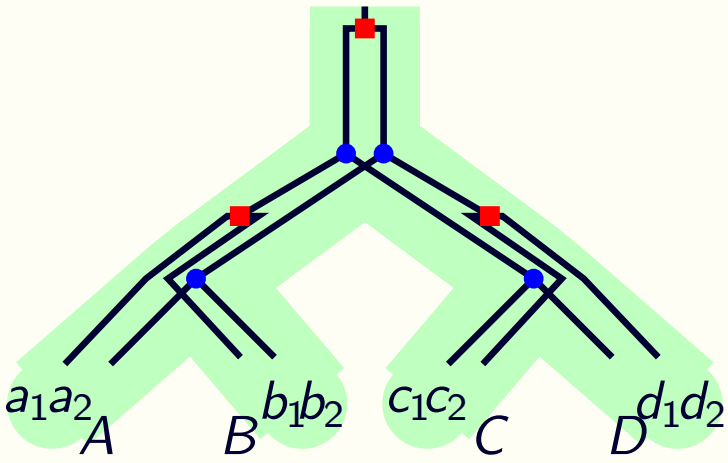
Extending to transfer networks

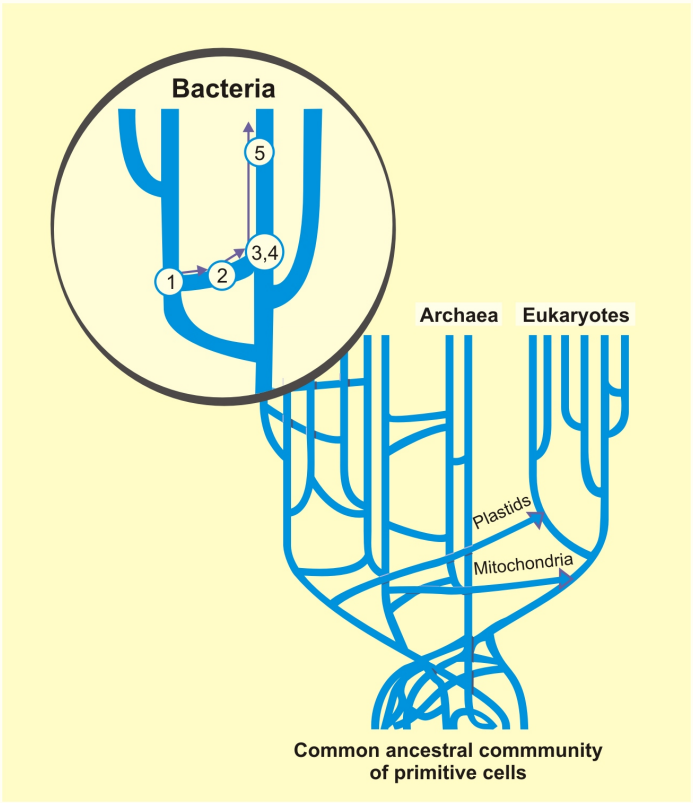




- We have been assuming that all gene propagation occurs in a species **tree**.
- In practice, **lateral gene transfer** can occur between branches.
- Open questions: Can we solve satisfy constraint graphs if the DS-tree must be "embedded" in an **acyclic network**

Extending to transfer networks





- We have been assuming that all gene propagation occurs in a species **tree**.
- In practice, **lateral gene transfer** can occur between branches.
- Open questions: Can we solve satisfy constraint graphs if the DS-tree must be "embedded" in an **acyclic network**

Experiments: Test instances

- Generated a set of connstraint graphs using data from ensembl.org.
- Chose 265 gene families of vertebrates with more than 20 genes from *Ensembl*.
- For each family, ProteinOrtho was ran to generate orthology relation graphs.
- By running ProteinOrtho with two different parameter settings (stricter vs looser characterization of orthology), a constraint graph is derived. (Unknown edges = edges marked as orthologs under one parameter and paralogs under another).
- (Same data set as used by Lafond & El-Mabrouk.)

Experiments: Test instances

- We ran our algorithm on the generated test instances.
- The hope: species trees returned by our algorithm could be reconciled to produce the known species tree.
- Unfortunately this was not the case :(
- DS-trees and species trees returned by our algorithm were very "flat", difficult to combine into a full picture.
- We believe this is partially due to small data size; hope to do better on a larger data set.

Experiments: Test instances

- We ran our algorithm on the generated test instances.
- The hope: species trees returned by our algorithm could be reconciled to produce the known species tree.
- Unfortunately this was not the case :(
- DS-trees and species trees returned by our algorithm were very "flat", difficult to combine into a full picture.
- We believe this is partially due to small data size; hope to do better on a larger data set.

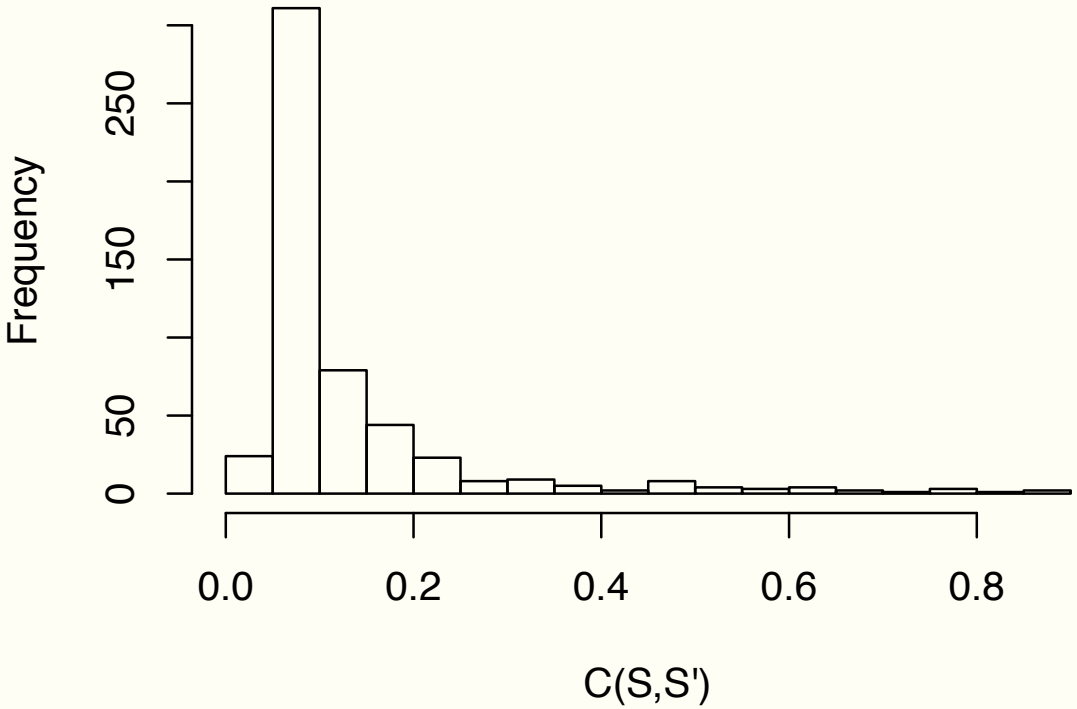
Experiments: Test instances

- We ran our algorithm on the generated test instances.
- The hope: species trees returned by our algorithm could be reconciled to produce the known species tree.
- Unfortunately this was not the case :(
- DS-trees and species trees returned by our algorithm were very "flat", difficult to combine into a full picture.
- We believe this is partially due to small data size; hope to do better on a larger data set.

Experiments: Test instances

- We ran our algorithm on the generated test instances.
- The hope: species trees returned by our algorithm could be reconciled to produce the known species tree.
- Unfortunately this was not the case :(
- DS-trees and species trees returned by our algorithm were very "flat", difficult to combine into a full picture.
- We believe this is partially due to small data size; hope to do better on a larger data set.

Relative consistency withknown species tree



- Of the satisfiable constraint graphs, 534 out of 2385 were not consistent with known species tree S.
- For 533 of these, our algorithm found a species tree S' for which they *were* consistent.
- We estimated the discordancy between S and each S' by finding minimum of triplets of S that must be unsatisfied in a tree not contradicting S'.

Future Work

Aims for the future:

- Apply algorithm on larger data sets.
- Extend algorithm to account for gene transfers (currently being worked on...).
- Extend algorithm to allow for "partially known" species trees.

Thanks for listening!

Future Work

Aims for the future:

- Apply algorithm on larger data sets.
- Extend algorithm to account for gene transfers (currently being worked on...).
- Extend algorithm to allow for "partially known" species trees.

Thanks for listening!