Supplementary material

Proof of Proposition 3

It suffices to show that for any triple of distinct vertices u, v, w, $uv \in E(G)$ and $vw \in E(G)$ imply that $uw \in E(G)$. Let E_{uv}, E_{vw} and E_{wu} denote the set of edges on the unique $u-v$, $v-w$ and $w-u$ paths in T, respectively. In T, one can get from w to u by first going from w to v, then from v to u. This implies that $E_{wu} \subseteq E_{vw} \cup E_{uv}$. As none of the edges in $E_{vw} \cup E_{uv}$ is marked as divergent, by definition, we have that u and w are similar, and thus uw is an edge of G.

An algorithm for STUC with allowable splits

We describe how to solve the STUC problem with allowable splits using dynamic programming. Given $X \subseteq \Sigma$, a set of clusters $\mathscr{C}$ and allowable splits $\mathbb{A}$, denote by $OPT(X, \mathscr{C}, \mathbb{A})$ the minimum number of losses in a species tree S, but only counting the lost clades that are strict subsets of X, and subject to the conditions that (1) $X \in clades(S)$ and (2) S has only splits from $\mathbb{A}$. To rephrase more formally, let $l_S(C, X)$ be the number of maximal C-free nodes v of the species tree S_C with the additional condition that $clade(v) \subset X$. Then $OPT(X, \mathscr{C}, \mathbb{A})$ is the minimum of $\sum_{C \in \mathscr{C}} l_S(C, X)$ over all possible species trees that contain the X clade (and only splits from $\mathbb{A}$). Define $OPT(X, \mathscr{C}, \mathbb{A}) = \infty$ if no such species tree exists. Clearly, $OPT(\Sigma, \mathscr{C}, \mathbb{A})$ yields a solution to our restricted STUC instance, since every loss occurs at a clade that is a strict subset of Σ, and any species tree contains the Σ clade. We would like to devise a recurrence formula for $OPT(\Sigma, \mathscr{C}, \mathbb{A})$.

Towards this goal, first let $\{A, B\}$ be a split of $X \subseteq \Sigma$ and let $C \in \mathscr{C}$. Define the cost $l_{A,B}(C)$ of C with respect to $\{A, B\}$ as

$$l_{A,B}(C) = \begin{cases} 1 & \text{if } C \cap (\Sigma \setminus X) \neq \emptyset, \text{ and exactly one of} \\ & C \cap A \text{ or } C \cap B \text{ is empty} \\ 0 & \text{otherwise} \end{cases}$$

In words, we count a loss for $l_{A,B}(C)$ is C has something outside $X = A \cup B$, and C loses all of A or all of B. We can now state our recurrence:

Lemma 1. *If $|X| = 1$, then $OPT(X, \mathscr{C}, \mathbb{A}) = 0$. If $|X| > 1$, then either $OPT(X, \mathscr{C}, \mathbb{A}) = \infty$, or $OPT(X, \mathscr{C}, \mathbb{A})$ is equal to*

$$\min_{\{A,B\} \in spl(X) \cap \mathbb{A}} \left(OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} l_{A,B}(C) \right)$$

where $spl(X)$ is the set of all possible splits of X.

Proof. We prove the Lemma by induction on $|X|$. As a base case, suppose $|X| = 1$. Then for any species tree S and any cluster C, $l_S(C, X)$ must be zero since S does not contain a C-free node v such that $clade(v) \subset X$ (as otherwise, $clade(v) = \emptyset$). Now suppose that the Lemma holds for any strict subset of X, and assume $OPT(X, \mathscr{C}, \mathbb{A}) \neq \infty$. Let S be a species tree containing a node v satisfying $clade(v) = X$ and having every split in $\mathbb{A}$, chosen such that S minimizes $\sum_{C \in \mathscr{C}} l_S(C, X)$. Let v_1, v_2 be the children of v and let $\{A, B\} = \{clade(v_1), clade(v_2)\}$. Then $\{A, B\} \in spl(X) \cap \mathbb{A}$. Since in the recurrence, the range of splits over which we take the minimum contains $\{A, B\}$, it suffices to prove that $OPT(X, \mathscr{C}, \mathbb{A}) = OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} l_{A,B}(C)$.

For some $C \in \mathscr{C}$, let F_A^C (resp. F_B^C) be the set of maximal C-free nodes w of S_C such that $clade(w) \subset A$ (resp. $clade(w) \subset B$). Moreover, let F^C be the set of maximal C-free nodes that belong to $\{v_1, v_2\}$. Then $OPT(X, \mathscr{C}, \mathbb{A}) = \sum_{C \in \mathscr{C}} l_S(C, X) = \sum_{C \in \mathscr{C}} (|F_A^C| + |F_B^C| + |F^C|)$. To prove the Lemma, we will show that $\sum_{C \in \mathscr{C}} |F_A^C| = OPT(A, \mathscr{C}, \mathbb{A})$, $\sum_{C \in \mathscr{C}} |F_B^C| = OPT(B, \mathscr{C}, \mathbb{A})$ and $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$.

To achieve this, we first show that the subtrees of S rooted at v_1 and v_2 must incur $OPT(A, \mathscr{C}, \mathbb{A})$ and $OPT(B, \mathscr{C}, \mathbb{A})$ losses, respectively. Consider a species tree S' that contains a node v' satisfying $clade(v') = A$, that has every split in $\mathbb{A}$, and such that $\sum_{C \in \mathscr{C}} l_{S'}(C, A) = OPT(A, \mathscr{C}, \mathbb{A})$. Note that S' exists since, in particular, S is a possible solution. Let $S'_{v'}$ be the subtree of S' rooted at v'. In S, replace the S_{v_1} subtree rooted at v_1 by the $S'_{v'}$ subtree. Observe that this new tree obtained has cost $OPT(A, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} |F_B^C| + \sum_{C \in \mathscr{C}} |F^C|$, implying that $\sum_{C \in \mathscr{C}} |F_A^C| = OPT(A, \mathscr{C}, \mathbb{A})$. By a similar reasoning, $\sum_{C \in \mathscr{C}} |F_B^C| = OPT(B, \mathscr{C}, \mathbb{A})$.

So far, we have shown that $OPT(X, \mathscr{C}, \mathbb{A}) = OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} |F^C|$. It remains to argue that $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$. Fix some $C \in \mathscr{C}$. If $s(C) < v$ or $s(C) \oplus v$, then neither of v_1 and v_2 can be a maximal C-free node. If $s(C) = v$, the same holds, because $s(C) = v$ implies that C contains an element from both $clade(v_1)$ and $clade(v_2)$. In both these cases, $F^C = \emptyset$. Accordingly, $l_{A,B} = 0$ in these cases (because when $s(C) \leq v$, we have $C \cap (\Sigma \setminus X) = \emptyset$, and when $s(C) \oplus v$, both $C \cap A$ and $C \cap B$ are empty). Assume that $s(C) > v$. It follows that $C \cap (\Sigma \setminus X) \neq \emptyset$, and also that $C \cap X \neq \emptyset$. The latter implies that v_1 and v_2 cannot both be maximal C-free nodes. If only v_1 is C-free maximal, then $F^C = \{v_1\}$. Accordingly, $l_{A,B}(C) = 1$, since $C \cap A = \emptyset$. The same holds if only v_2 is maximal C-free. Finally, if none of v_1 and v_2 are maximal C-free, then $F^C = \emptyset$, and accordingly $l_{A,B}(C) = 0$. In every case, we have equality between $|F^C|$ and $l_{A,B}(C)$. This shows that $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$, concluding the proof. □

Observe that $spl(X) \cap \mathbb{A}$ can be computed in time $O(|\mathbb{A}||X|)$, as it suffices to iterate over every element of $\mathbb{A}$ and check whether the split partitions X.

In the next section, we prove Theorem 7, i.e. that the above dynamic programming formulation leads to an algorithm that runs in time $O(|\mathbb{A}|^2 |\mathscr{C}||\Sigma|)$.

We construct the set $\mathbb{A}$ in a heuristic fashion as follows. We build a collection D of subsets of Σ, and $\mathbb{A}$ consists of all the pairs $\{D_1, D_2\}$ where $D_1, D_2 \in D$ and $D_1 \cap D_2 = \emptyset$. To construct D, sort the species of Σ by non-increasing order of *frequency*, where the frequency of x is the number of clusters of $\mathscr{C}$ that contain x. This results in an ordering $(s_1, \ldots, s_m)$ of Σ. Then for any $i, j \in [m]$ with $i < j$, add the set $\{s_i, s_{i+1}, \ldots, s_j\}$ to D. For good measure, we also add every cluster $C_i \in \mathscr{C}$ to D, concluding the construction. The idea behind the ordering is that if a set of species have a similar frequency, it is possibly because they always appear together in the same clusters. This set may thus be part of some split of S.

Proof of Theorem 7

Consider the algorithm that starts computing $OPT(\Sigma, \mathscr{C}, \mathbb{A})$ and recursively computes $OPT(X, \mathscr{C}, \mathbb{A})$ for the necessary values of X according to the formulation of Lemma 1. Assume that $OPT(X, \mathscr{C}, \mathbb{A})$ is computed once for each value of X, and stored in memory if this value is required later on. The value of $OPT(X, \mathscr{C}, \mathbb{A})$ only needs to be computed if $\mathbb{A}$ contains a pair of the form $\{X, Y\}$. Therefore, at most $2|\mathbb{A}|$ computations of $OPT(X, \mathscr{C}, \mathbb{A})$ are necessary. The time required to compute $OPT(X, \mathscr{C}, \mathbb{A})$, without counting the time for the recursive calls, corresponds to the time required to compute $l_{A,B}(C)$ for each $C \in \mathscr{C}$ and each $\{A, B\} \in spl(X) \cap \mathbb{A}$. The value $l_{A,B}(C)$ only needs to compute intersections over subsets of Σ and thus takes time $O(|\Sigma|)$. Since there are at most $|\mathbb{A}|$ splits in $spl(X) \cap \mathbb{A}$, the time required for

$OPT(X, \mathscr{C}, \mathbb{A})$ is $O(|\mathbb{A}||\mathscr{C}||\Sigma|)$. As mentioned before, X can take $2|\mathbb{A}|$ values, and the overall complexity is $O(|\mathbb{A}|^2|\mathscr{C}||\Sigma|)$.

Proof of Theorem 8

First note that if $\sigma(x) = \sigma(y)$, then no edge will ever be added between x and y, due to the condition in line 5. Hence no two genes from the same species are ever made orthologs. We must prove that the graph output by the algorithm has no forbidden P_3 and no P_4. Beforehand we make an important observation. Consider an iteration, when some cluster C is merged with its favorite cluster C_j. Let $z \in C_j$. Then either z becomes orthologous with every member of C, or with none of them. Moreover, these relations between z and C remain until the end of the algorithm.

Now, suppose that vertices x, y, z introduce a P_3 in G (with edges xy and yz) at the i-th iteration, for some $i \in [k-1]$. Assume that $\sigma(x), \sigma(y), \sigma(z)$ are all distinct. We must prove that $\sigma(x)\sigma(z)|\sigma(y)$ is a triplet of S, as required by S-consistency. Let the clusters $C_1, \ldots, C_k$ be the clusters at the beginning of the i-th iteration (in particular, there are $i-1$ empty clusters due to line 7). To avoid ambiguity, if we need to refer to some cluster C_l at the beginning of the h-th iteration, we will write $C_l^{(h)}$ (hence, under our notation, we use C_l as a shorthand for $C_l^{(i)}$). Now, let C be the cluster chosen at the i-th iteration, and let C_j be the favorite cluster of C chosen at this iteration. Since only edges between C and C_j are added at this i-th iteration, C and C_j must both contain one of $\{x, y, z\}$.

Assume that x, y and z all belong to a distinct cluster at the i-th iteration, say the clusters C, C_j and C_h. Then there is an edge going from C_h to at least one of C or C_j. This edge was added in an iteration prior to i, implying that one of C, C_j or C_h must have been emptied, a contradiction. We may therefore assume that C and C_j contain all of $\{x, y, z\}$. We check all the possible locations of x, y and z:

- If $x, z \in C$ and $y \in C_j$, then $\sigma(y) \oplus s(C)$ (because every $y-C$ edge must have been added by the algorithm). Moreover, $\sigma(x), \sigma(z) \leq s(C)$ (by the definition of $s(C)$). This implies $\sigma(x)\sigma(z)|\sigma(y)$ is a triplet of S, as required.
- If $x, y \in C$ and $z \in C_j$, z is a neighbor of y but not x. But z should be a neighbor of every member of C, or of none of them. Hence z cannot form a P_3 with x and y. Similarly, we cannot have $z, y \in C$ and $x \in C_j$.
- If $x \in C$, $y, z \in C_j$, then $\sigma(y) \oplus s(C)$, $\sigma(x) \leq s(C)$ and $\sigma(z) \leq s(C)$ (otherwise zx would be an edge). This implies the $\sigma(x)\sigma(z)|\sigma(y)$ triplet in S, as required. The same idea works if $z \in C$ and $x, y \in C_j$.
- Finally, suppose that $y \in C$ and $x, z \in C_j$. Then $\sigma(x) \oplus s(C)$ and $\sigma(z) \oplus s(C)$. For the sake of contradiction, assume w.l.o.g. that $\sigma(y)\sigma(x)|\sigma(z)$ is a triplet of S (the other contradicting triplet is $\sigma(y)\sigma(z)|\sigma(x)$, and the same proof applies). Since x and z do not share an edge, they did not belong to the initial set of given clusters. Then there must have been some iteration $h < i$ such that $C_j^{(h)}$ was the favorite of some cluster $C_l^{(h)}$, with either $x \in C_j^{(h)}, z \in C_l^{(h)}$ or vice-versa. Suppose $x \in C_j^{(h)}, z \in C_l^{(h)}$. As no edge was added between x and z, we had $\sigma(x) \leq s(C_l^{(h)})$. Thus $s(C_l^{(h)})$ is a common ancestor of $\sigma(x)$ and $\sigma(z)$. Since $\sigma(y)\sigma(x)|\sigma(z)$ is a triplet of S and $\sigma(x), \sigma(z) \oplus s(C)$, we have $s(C_l^{(h)}) > s(C)$. However, the algorithm iterates over the clusters in non-decreasing order with respect to S, a contradiction since C should have been handled before C_h at the h-th iteration. The same argument holds for the case $z \in C_j^{(h)}, x \in C_l^{(h)}$.

This proves that every P_3 satisfies the requirement of Theorem 6.

Now, suppose that vertices w, x, y, z introduce a P_4 at the i-th iteration (with edges wx, xy, yz), at which point C is merged with its favorite C_j. Observe that as above, we may assume that C and C_j contain all of $\{w, x, y, z\}$. Notice also that C_j cannot contain only one vertex from this P_4: this vertex must either be adjacent to the 3 others, or to none of them, both cases preventing a P_4. Also, C_j cannot contain exactly two of these vertices. Indeed, if $w, x \in C_j$ and $y, z \in C$, then since xy is an edge, x shares an edge with every member of C. This includes the xz edge, a contradiction. One can check that the type of argument holds for the other five cases $w, y \in C_j$, $w, z \in C_j$, $x, y \in C_j, x, z \in C_j$ and $y, z \in C_j$ in which C_j has exactly two vertices of the P_4.

It follows that C_j must have 3 vertices. In particular, we either have both $x, z \in C_j$, or only one of x or z is in C_j. Suppose $x, y \in C_j$ and w.l.o.g. $w \in C, z \in C_j$. Then $\sigma(y) \leq s(C)$, $\sigma(z) \leq s(C)$ and $\sigma(x) \oplus s(C)$. Note that this implies $\sigma(x) \neq \sigma(z)$. Moreover, $\sigma(x) \neq \sigma(y) \neq \sigma(z)$, since xy and yz are edges. These fact imply that $\sigma(y)\sigma(z)|\sigma(x)$ is a triplet of S, and that the x, y, z P_3 should be forbidden, contradicting the fact that every P_3 is consistent. It remains to cover the case when only one of x or y is in C_j. Suppose, w.l.o.g. that $x \in C$, $w, y, z \in C_j$. Note that $\sigma(w) \oplus s(C)$, $\sigma(y) \oplus s(C)$ and $\sigma(z) \leq s(C)$. As we did before, consider the iteration h at which w and z both ended up in C_j. Then either $w \in C_j^{(h)}$ and $z \in C_l^{(h)}$ for some l, or vice-versa. If $w \in C_j^{(h)}$ and $z \in C_l^{(h)}$, then $\sigma(w) \leq s(C_l^{(h)})$. Thus $C_l^{(h)}$ is a common ancestor of $\sigma(w)$ and $\sigma(z)$, which implies $s(C_l^{(h)}) > s(C)$ since $\sigma(w) \oplus s(C)$ and $\sigma(z) \leq s(C)$. This contradicts the ordering of the iterations of the algorithm. If $z \in C_j^{(h)}$ and $w \in C_l^{(h)}$, then $\sigma(z) \leq s(C_l^{(h)})$, $s(C_l^{(h)})$ is a common ancestor of $\sigma(w)$ and $\sigma(z)$, and we reach the same contradiction. This concludes the proof.