

RESEARCH

Open Access

Orthology and paralogy constraints: satisfiability and consistency

Manuel Lafond*, Nadia El-Mabrouk

From Twelfth Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics

Cold Spring Harbor, NY, USA. 19-22 October 2014

Abstract

Background: A variety of methods based on sequence similarity, reconciliation, synteny or functional characteristics, can be used to infer orthology and paralogy relations between genes of a given gene family \mathcal{G} . But is a given set \mathcal{C} of orthology/paralogy constraints possible, i.e., can they simultaneously co-exist in an evolutionary history for \mathcal{G} ? While previous studies have focused on full sets of constraints, here we consider the general case where \mathcal{C} does not necessarily involve a constraint for each pair of genes. The problem is subdivided in two parts: (1) Is \mathcal{C} satisfiable, i.e. can we find an event-labeled gene tree G inducing \mathcal{C} ? (2) Is there such a G which is *consistent*, i.e., such that all displayed triplet phylogenies are included in a species tree?

Results: Previous results on the *Graph sandwich problem* can be used to answer to (1), and we provide polynomial-time algorithms for satisfiability and consistency with a given species tree. We also describe a new polynomial-time algorithm for the case of consistency with an unknown species tree and full knowledge of pairwise orthology/paralogy relationships, as well as a branch-and-bound algorithm in the case when unknown relations are present. We show that our algorithms can be used in combination with ProteinOrtho, a sequence similarity-based orthology detection tool, to extract a set of robust orthology/paralogy relationships.

Introduction

Gene families, usually constructed from sequence similarity, group *homologous* genes, i.e., genes sharing a common ancestry: starting from a single gene copy, a history of speciations, duplications and losses is assumed to be at the origin of the observed set of extant genes. Deciphering the *orthology* (divergence following a speciation) and *paralogy* (divergence following a duplication) relations between pairs of genes inside a gene family is important and lies at the heart of many genomics studies. The reconstruction of species trees for example is usually based on the selection and alignment of orthologous gene copies. From a functional point of view, orthologs are believed to be more likely similar in function than paralogs [1]. Orthology/paralogy information is often derived from a reconciliation approach (first introduced

by Goodman in 1979 [2]). A gene tree that best reflects the evolution of the sequences is first constructed for the gene family. Assuming a known phylogeny for the set of taxa, the non-agreement between the two trees is then explained by a set of duplication and loss events (other events such as horizontal gene transfer might also be inferred by reconciliation, although we will ignore them here). Reconciliation leads to a labeling of internal nodes of the gene tree as duplication/speciation nodes, yielding a full orthology/paralogy interpretation for each pair of genes (cf. e.g. TreeFam [3,4] used for constructing the *Ensembl Compara* gene trees [5], PHOG [6], MetaPHOrs [7]). This approach assumes that an accurate gene tree can be constructed for the gene family. Although inferring phylogenies is a field with a very long history, due to various limitations constructing good gene trees is still challenging. A variety of other methods have been developed for the purpose of orthology/paralogy detection. A well-known class of algorithms is based on clustering

* Correspondence: lafonman@iro.umontreal.ca
Department of Computer Science and Operational Research, University of Montreal, Chemin de la Tour, H3C3J7 Montreal, Canada

genes according to their sequence similarity (cf. e.g. the COG database [8], OrthoMCL [9], InParanoid [10], Proteinortho [11], orthoDB [12], eggNOG [13]). Recently, we investigated another way of detecting orthology/paralogy based on conserved syntenies (conservation in gene order) [14,15]. Other initiatives, such as the Gene Ontology project [16], provide functional annotation that can be used as another source of orthology relations. In contrast to the reconciliation approach, only partial relations can be expected from such tree-free methods.

The orthology/paralogy information suggested by gene tree reconciliation may be contradictory with that suggested by an external source. As gene trees are known to be error-prone, more confidence can be given to such homology information when it is well-supported by various genomic observations. This raises the problem of gene tree editing based on a known set \mathcal{C} of pairwise orthology/paralogy constraints. But prior to any algorithmic consideration, one should be able to state whether the set \mathcal{C} is possible, i.e. whether all constraints can simultaneously co-exist in an evolutionary history of the gene family. In a recent work [14], we showed that a set of orthology constraints is always possible and we gave a polynomial-time algorithm for correcting a gene tree in a minimal way according to the Robinson-Foulds distance.

Recent studies have considered the connection of trees and orthology from the angle of reconstructing phylogenies from orthology relations [17-19]: How much information about the gene tree, the species tree and their reconciliation is already contained in the orthology relation between genes? In other words, having a set \mathcal{C} of full pairwise orthology/paralogy relations (each pair of genes is constrained), can one reconstruct the gene and species trees? Similarly to gene tree editing, the first question to be asked is whether the orthology/paralogy constraints can simultaneously co-exist in a history of the gene family. Interestingly, by making the link with symbolic ultrametrics and co-graphs, a simple characterization of *satisfiability* (symbolic ultrametric) for full paralogy/orthology relations is given in [17], where satisfiability relates to the existence of an event-labeled gene tree G (symbolic representation) leading to \mathcal{C} . Notice that satisfiability does not mean the possibility for orthology/paralogy relations to co-exist in a true history, as the triplet phylogenies contained in G are not necessarily *consistent* (included in a species tree). The derivation of a species tree from an event-labeled gene tree is considered in [18]. Finally, the outline of a computational framework for the construction of a least resolved species tree S from a set of orthology/paralogy relations, involving the extraction of maximum satisfiable relations and maximum consistent triple set is given in [19].

Here, we consider the general case for \mathcal{C} : in contrast with [17,18], we do not require \mathcal{C} to be full, i.e., to involve

a constraint for each pair of genes. We introduce the notations and problems in the following section. We then show how previous results on the *Graph sandwich problem* can be used to solve the problem of satisfiability. The developed algorithm for satisfiability is then adapted to the problem of consistency with a given species tree. In the case of an unknown species tree, we then study the problem of finding a gene tree that is consistent with *some* species tree. Finally we show in the result section how our methodology can be used, in combination with PROTEINORTHO[11], a sequence similarity-based orthology detection tool, to extract a set of robust orthology/paralogy relationships.

Notations and problem statement

In the rest of this paper, we consider a set Σ of species and a gene family \mathcal{G} where each gene x belongs to a species $s(x)$ of Σ . We generalize the notation s to subsets of genes: if $U \subset \mathcal{G}$, $s(U) = \{s(x) : x \in U\}$.

As we only consider rooted trees, we will sometimes omit the word “rooted”. Let T be a tree. We denote by $r(T)$ its root and by $L(T)$ its set of leaves. For any internal node x of T , we denote by T_x the subtree of T rooted at x . We say that a node y is an *ancestor* of x if the two nodes belong to the same path from a leaf to the root of T , and y is closer to the root. Two nodes x and y are *unrelated* if $x \neq y$ and none is the ancestor of the other. For a set of leaves $U \subseteq L(T)$, we denote by $lca_T(U)$ the *least common ancestral node* of U in T , i.e. the common ancestral node of the elements of U which is the farthest from the root.

Let L' be a subset of $L(T)$. The *restriction* $T|_{L'}$ of T to L' is the tree with leaf set L' obtained from $T_{lca_T(L')}$ by removing all leaves that are not in L' , and all internal nodes of degree 2, except the root. Let T' be a tree such that $L(T') = L' \subset L(T)$. We say that T *displays* T' iff $T|_{L'}$ is label-isomorphic to T' .

A triplet is a binary tree on a set L with $|L| = 3$. For $L = \{x, y, z\}$, we denote by xyz the unique triplet t on L with root $r(t)$ for which $lca_t(x, y) \neq r(t)$ holds. We denote by $tr(T) = \{T|_L : L \in \binom{L(T)}{3} \text{ and } T|_L \text{ is binary}\}$ the set of all rooted triplets of a tree T .

Evolution of species and genes: A *species tree* S for Σ is a rooted tree whose leaves are in bijection with Σ , representing the evolutionary relationships between the species: an internal node is an ancestral species at the moment of a speciation event, and its children are the descendants. Although species trees are generally binary, we do not make this assumption here. Genes of \mathcal{G} undergo speciation when the species to which they belong do. Within a species, genes can be duplicated or lost. A *history* H for \mathcal{G} is a tree representing the evolution of the gene family through speciations and duplications: each leaf of H is labeled by a

gene of \mathcal{G} , and each internal node refers to an ancestral gene at the moment of an event (speciation or duplication). Therefore each internal node of H can be labeled as a speciation (*Spec*) or duplication (*Dup*) event.

As H is a history “embedded” in the species tree S of Σ , it must reflect a speciation history *consistent* with S : any speciation node of H should reflect a clustering of species in agreement with S . To formally define consistency, let first introduce a more general set of labeled trees. We call a *DS-tree* for \mathcal{G} a pair (G, ℓ) , where G is a tree with $L(G) = \mathcal{G}$, and ℓ is a function $\ell : V(G) \setminus L(G) \rightarrow \{Dup, Spec\}$ labeling each internal node of G as a duplication or a speciation node. For simplicity, we often refer to G as the *DS-tree* for \mathcal{G} without explicitly stating ℓ , and assume the internal nodes of G are labeled *Dup* or *Spec*. For some $X \subseteq L(G)$, we implicitly assume that the internal nodes of $G|_X$ share the same label as in G .

Definition 1 Let G be a DS-tree for \mathcal{G} and S be a species tree for Σ . We say that G is consistent with S if and only if, for any speciation node x of G and any two children y, z of x , $lca_S(s(L_y))$ and $lca_S(s(L_z))$ are unrelated in S , where L_y and L_z are the leaf-sets of G_y and G_z respectively.

Now a history H for \mathcal{G} is simply a DS-tree for \mathcal{G} consistent with the species tree S for Σ . Denote

$tr_S(G) = \{s(x)s(y)|s(z) : xy|z \in tr(G) \text{ is rooted at a speciation and } s(x) \neq s(y)\}$

The triplets of $tr_S(G)$ are called *speciation triplets*. The following theorem, which is a reformulation of Theorem 6 in [18], relates consistency to speciation triplets.

Theorem 1 Let G be a DS-tree for \mathcal{G} and S be a species tree for Σ . Then G is consistent with S if and only if S displays all triplets of $tr_S(G)$.

From the Fitch [20] terminology, two leaves x and y of a history are *orthologous* if $lca_H(x, y)$ is a speciation node, and *paralogous* otherwise. We extend this terminology to a more general DS-tree.

Definition 2 Let G be a DS-tree for \mathcal{G} . Then two genes x, y of G are *orthologous with respect to (w.r.t.) G* if $lca_G(x, y)$ is a speciation node, and *paralogous w.r.t. G* if $lca_G(x, y)$ is a duplication node.

Therefore a DS-tree induces a set of orthology/paralogy relationships between genes.

Constraint graph: A constraint is simply an unordered pair of genes $\{x, y\} \in \binom{\mathcal{G}}{2}$. A set of orthology/paralogy constraints on \mathcal{G} (or simply a constraint set) is a pair $C = (C_O, C_P)$ of subsets $C_O, C_P \subset \binom{\mathcal{G}}{2}$ such that $C_O \cap C_P = \emptyset$. C_O represents the *orthology constraints* and C_P the *paralogy constraints*. We say that C is a *full constraint set* if

\mathcal{G} . For example, a history H for \mathcal{G} induces a full constraint set.

We represent a constraint set $C = (C_O, C_P)$ by an edge-bicoloured undirected graph $R = (V, E, U)$, called a *constraint graph*, with vertex set $V = \mathcal{G}$, and two edge sets $E = C_O$ and $U = \binom{\mathcal{G}}{2} \setminus (C_O \cup C_P)$. Said differently, two genes are linked by an edge of E if they are constrained by orthology, unlinked if they are constrained by paralogy, and linked by a “special” edge of U if the relation between the two genes is unknown. We refer to the edges of E as the *orthology edges*, to those in U as the *unknown edges* and we refer to the unlinked pairs of genes as the *paralogy non-edges*. An example of a partial constraint graph is given in Figure 1.

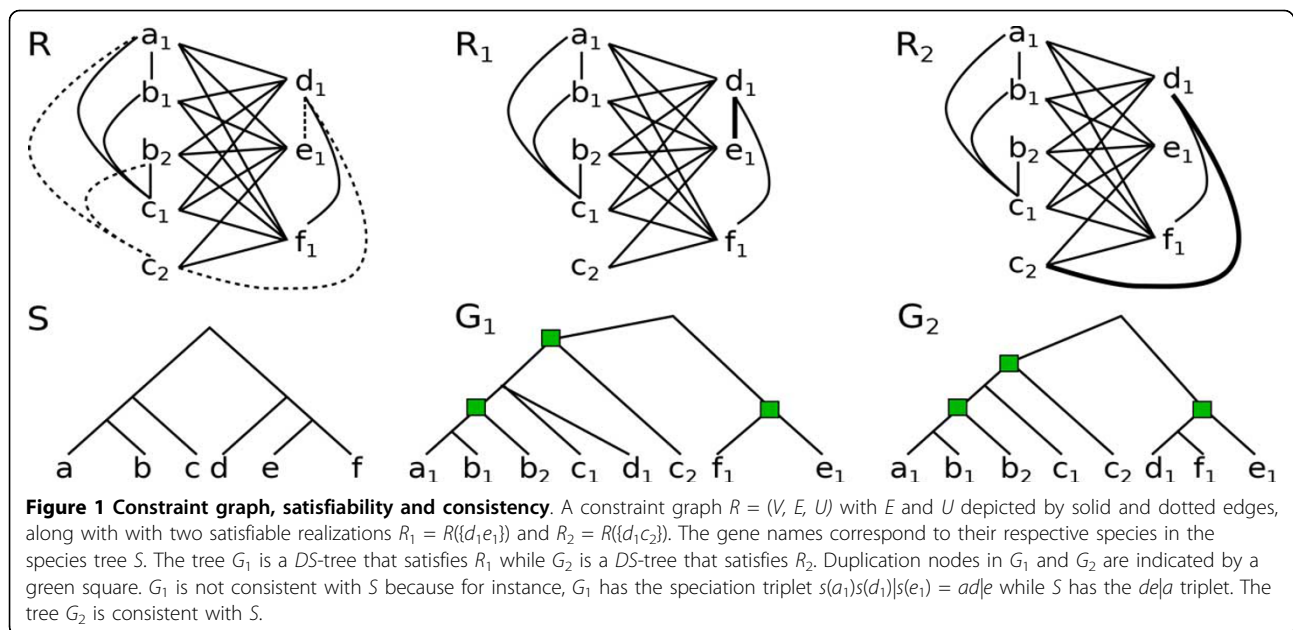
If C is a full constraint set then $U = \emptyset$. R is called a *full constraint graph* in this case. The *complement* of R is the graph \bar{R} obtained by the alternative choice of linking paralogues instead of orthologues. Formally, $\bar{R} = (V, \bar{E}, U)$, where \bar{E} is the complement of E defined by $e \in \bar{E}$ iff $e \notin E$. Notice that R and \bar{R} share the same set U of unknown edges. We denote by $R[X]$ the graph R induced by $X \subseteq V$, i.e. $R[X] = (X, E(X), U(X))$ where $E(X)$ (resp. $U(X)$) are the edges of E (resp. U) having both endpoints in X . Note that if $U = \emptyset$, $R[X]$ corresponds to the usual definition of the graph induced by X .

Satisfiability and consistency Given a constraint set C (or similarly a constraint graph R), is C *possible*, i.e. can we find a history for \mathcal{G} inducing the orthology and paralogy constraints of C ? As an orthology constraint for two genes belonging to the same genome cannot be induced by a history for \mathcal{G} , we assume in the rest of this paper that the set $\tilde{C}_P = \{\{x, y\} \in \binom{\mathcal{G}}{2} : s(x) = s(y)\}$ is included in C_P . A *trivial set of paralogy constraints* is a set C_P restricted to \tilde{C}_P .

The question whether C is possible is in two parts: (1) is C *satisfiable*, i.e. can we find a DS-tree G inducing C and (2) is there such a G which is consistent with a species tree? Formal definitions follow.

Definition 3 Let $R = (V, E, U)$ be a constraint graph and G be a DS-tree with $L(G) = V$. We say that G *satisfies R* if for two genes $x, y \in L(G)$, if $xy \in E$ then they are orthologous w.r.t. G , and if $xy \in E \setminus U$ then they are paralogous w.r.t. G . We say that R is *satisfiable* if there exist a DS-tree G that satisfies R .

If $U = \emptyset$, then R being satisfiable means that we can make a choice for the unknown edges as orthology edges and paralogy non-edges to obtain a full constraint graph that is satisfiable. For $F \subset U$, the *realization* of R by F corresponds to choosing F as orthology edges, and $U \setminus F$ as paralogy non-edges, leading to the full constraint



graph $R(F) = (V, E \cup F, \emptyset)$. We call $R(\emptyset)$ and $R(U)$ the *empty* and *full* realizations, respectively.

As a history is a DS-tree, a set of constraints that is not satisfiable is clearly not possible, i.e. there is no history that depicts the orthology/paralogy relationships given by the constraints. Moreover, satisfiability is not sufficient to ensure the possibility of such an history, as a DS-tree is not always consistent with a species tree. Figure 1 shows an example of a constraint graph R along with two satisfying realizations R_1 and R_2 . However, R_1 cannot be made consistent with a given species S whereas R_2 can.

Definition 4 Let R be a constraint graph for \mathcal{G} . We say that R is consistent with a species tree S if and only if there is a realization of R which is satisfiable by a DS-tree G which is consistent with S . More generally, we say that R is consistent if and only if there is a species tree S such that R is consistent with S .

The three following sections are respectively dedicated to the three following questions: (1) Given a constraint graph $R = (V, E, U)$, is R satisfiable? (2) Given a satisfiable constraint graph $R = (V, E, U)$, and a species tree S , is R consistent with S ? (3) Given a satisfiable constraint graph $R = (V, E, U)$, is R consistent, i.e. can we find a species tree S such that R is consistent with S ?

Satisfiability of a constraint graph

The problem of constraint graph satisfiability has been addressed in [17] in the restricted case of a full set of constraints. The following theorem is a reformulation of one of the main results of this paper.

Theorem 2 ([17]) *A full constraint graph R is satisfiable if and only if R is P_4 -free (or equivalently, iff \bar{R} is P_4 -free since P_4 is self-complementary), meaning that no four vertices of R induce a path of length 4.*

Consider now the general case of a constraint graph $R = (V, E, U)$ with $U = \emptyset$. Then the problem is to find a realization $R(F)$ that is itself satisfiable, i.e. P_4 -free. It turns out that this problem is a reformulation of the well-known *Graph sandwich problem* for P_4 -free graphs. It can be stated as follows : given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, with $E_1 \subseteq E_2$, does there exist a P_4 -free graph $G = (V, E)$ such that $E_1 \subseteq E \subseteq E_2$. That is, G must contain every edge of G_1 and every non-edge of G_2 . It is then clear that this is equivalent to finding a P_4 -free realization of $R = (V, E_1, U = E_2 \setminus E_1)$. A $O(|V|^3)$ algorithm was proposed in [21] to solve this problem. In this section, we restate under our formalism some of the useful results of this paper, and give a modified version of the proposed algorithm that outputs a DS-tree that satisfies R whenever there is one. We will make use of the following well-known characterization of P_4 -free graphs.

Lemma 1 *A graph Γ is P_4 -free if and only if, for any subset X of vertices of Γ with $|X| \geq 2$, either $\Gamma[X]$ or $\overline{\Gamma[X]}$ is disconnected.*

The next lemmata establish an important *heritability* property on satisfiable graphs: every restriction $R[X]$ of R must be satisfiable for R to be satisfiable.

Lemma 2 *Let G be a DS-tree that satisfies a realization $R(F)$, for some $F \subseteq U$. Let $X \subseteq V$ and let $F_X = \{ab \in F : a, b \in X\}$. Then $G|_X$ is a realization of $R[X](F_X)$.*

Proof: Let $a, b \in X$. First observe that a and b have the same orthology/paralogy relationship in $R(F)$ and $R[X]$ (F_X). Let $c = lca_G(a, b)$. Now, c is also an internal node of $G|_X$, and moreover $c = lca_{G|_X}(a, b)$. As c has the same *Dup* or *Spec* label as in G , c correctly displays the relationship between a and b . Thus $G|_X$ satisfies every relationship in $R[X](F_X)$.

The heritability property is then immediate.

Lemma 3 [21] *A constraint graph $R = (V, E, U)$ is satisfiable if and only if for any $X \subseteq V$, $R[X]$ is satisfiable.*

Theorem 3 [21] *A constraint graph R is satisfiable if and only if at least one of the two following holds :*

1 $R(\emptyset)$ is disconnected, and all of its connected components are satisfiable;

2 $\overline{R(U)}$ is disconnected, and all of its connected components are satisfiable.

Proof: \Leftarrow For 1.: Suppose $R(\emptyset)$ is disconnected. Let $\{R_1, \dots, R_k\}$ be the connected components of $R(\emptyset)$ with $k > 1$, all being satisfiable. As each R_i is satisfiable, there is a *DS*-tree G_i that satisfies a realization $R_i(F_i)$ of R_i . Let $F = \cup_{1 \leq i \leq k} F_i$. Then the realization $R(F)$ of R is a full constraint graph with k full constraint components $R_i(F_i)$ in which no two components share an edge. In other words, there is a paralogy non-edge between each pair of genes from two different components. Therefore, joining the roots of G_1, \dots, G_k under a common duplication node yields a *DS*-tree for $R(F)$, which shows that R is satisfiable. The proof when 2. holds is the same, except that we root G at a speciation node since the components of $\overline{R(U)}$ are pairwise-complete in $R(U)$.

\Rightarrow Suppose that both conditions do not hold. If $R(\emptyset)$ or $\overline{R(U)}$ has a component that is not satisfiable, then by Lemma 3, R is not satisfiable. So instead suppose that each of $R(\emptyset)$ and $\overline{R(U)}$ has a single connected component. Let $F \subseteq U$. The realization $R(F)$ of R must be connected as $R(\emptyset)$ is already connected and $\overline{R(F)}$. $\overline{R(F)}$ must also be connected, as choosing all edges of U leaves $\overline{R(U)}$ connected and $\overline{ER(U)} \subseteq \overline{E(R(E))}$. Since both $R(F)$ and $\overline{R(F)}$ are connected, by Lemma 1 $R(F)$ is not P_4 -free, and thus not satisfiable by Theorem 2. As this is true for any realization $R(F)$ of R , i.e. for any $F \subseteq U$, it follows that R is not satisfiable.

Theorem 3 suggests the recursive algorithm BuildDSTree that begins by finding out if one of $R(\emptyset)$ or $\overline{R(U)}$ is disconnected. If so, it creates a node of the appropriate type with children being the identified components, and repeats the process on each such component.

Algorithm BuildDSTree ($R = (V, E, U), v$)

where R is a (possibly induced) constraint graph and v is the current node of G we are creating

IF $|V| = 1$; RETURN; $R(\emptyset) = (V, E, \emptyset)$

Find the connected components CC of $R(\emptyset)$ through a depth-first search

IF $|CC| > 1$;

type \leftarrow Dup

ELSE

$\overline{R(U)} = (V, \overline{E}, \emptyset)$

type \leftarrow Spec

Find the connected components CC of $\overline{R(U)}$ through a depth-first search IF $|CC| = 1$; output "Unsatisfiable", and halt the recursion

END IF

v.type \leftarrow type

FOR $C \in CC$;

Add child node v_C to v

BuildDSTree($R[C], v_C$)

END FOR

RETURN

If n is the number of genes in G , the algorithm creates a *DS*-tree G with at most $n - 1$ internal nodes (or stops before if R is unsatisfiable). For each such internal node v , the time taken to go through the algorithm is dominated by (at most) two depth-first searches that are performed on $L(G_v)$, and the rest of the work is handled by children nodes. So the time taken to handle v is bounded by the number of edges/non-edges in $R[L(G_v)]$, which is $O(|L(G_v)|^2) \subseteq O(n^2)$. Therefore the time-complexity of BuildDSTree is in $O(n^3)$.

Consistency with a given species tree

Let $R = (V, E, U)$ be a constraint graph for \mathcal{G} and S be a species tree for Σ . We want to know whether the orthology/paralogy constraints represented by R can be induced by a history for \mathcal{G} consistent with S . More precisely, is there a realization $R(F)$ of R that is satisfiable and such that the *DS*-tree satisfying $R(F)$ is consistent with S ? If R is not satisfiable, then the answer is clearly no. Therefore hereafter we assume that R is satisfiable. We first show that the problem at hand still has the heritability property.

Lemma 4 *R is consistent with S if and only if for any $X \subseteq V$, $R[X]$ is consistent with S .*

Proof: The ' \Leftarrow ' part is trivial since we can choose $X = V$ to show that R is consistent with S . Conversely, assume R is consistent with S . Let G be a *DS*-tree for some realization of R such that G is consistent with S , and let $X \subseteq V$. By Lemma 2, $G|_X$ is a *DS*-tree for $R[X]$. Let $ab|c \in tr_S(G|_X)$. Since going from $G|_X$ to G only involves adding subtrees on branches of $G|_X$, it follows that $ab|c \in tr_S(G)$. Therefore, $tr_S(G|_X) \subseteq tr_S(G)$. Now, since S displays $tr_S(G)$, $G|_X$ is a realization of $R[X]$ that is consistent with S .

We need to introduce one last notation before stating the main theorem for characterizing consistency of a

constraint graph R with a species tree S . Let $R(F)$ be a realization of R , and let $CC = \{R_1, \dots, R_k\}$ be the connected components of $\overline{R(F)}$. Notice that the components of CC are pairwise complete in $R(F)$. A *speciation partition* $P = \{P_1, \dots, P_{|P|}\}$ is a non-trivial partition of CC (i.e. $|P| > 1$) such that $lca_S(s(P_i))$ is unrelated to $lca_S(s(P_j))$ whenever $i \neq j$.

Theorem 4 R is consistent with S if and only if at least one of the following conditions holds:

1 $R(\emptyset)$ is disconnected and each connected component is consistent with S ;

2 $\overline{R(U)}$ is disconnected, its components admit a speciation partition and each component in this partition is consistent with S .

Proof: \Leftarrow For 1., Let $\{R_1, \dots, R_k\}$ be the connected components of $R(\emptyset)$, each R_i having a DS -tree G_i consistent with S . We can then join the roots of G_1, \dots, G_k under a common duplication parent. This yields a DS -tree G that satisfies R as each pair of components of $R(\emptyset)$ are related by paralogy. Furthermore, all rooted triplets of G that were not in any G_i are rooted at $r(G)$, a *Dup* node. Therefore, $tr_S(G) = \cup_{1 \leq i \leq k} tr_S(G_i)$, which S displays.

\Leftarrow 2.: Let $P = \{P_1, \dots, P_k\}$ be a non-trivial speciation partition of the connected components of $\overline{R(U)}$. By assumption every $P_i \in P$ has a DS -tree G_i that is consistent with S , implying that S displays $\cup_{1 \leq i \leq k} tr_S(G_i)$. Here all elements of P are components of R that are pairwise-complete, and we obtain a DS -tree G for R by joining G_1, \dots, G_k under a common speciation parent. Let $T = tr_S(G) \setminus \cup_{1 \leq i \leq k} tr_S(G_i)$. Every triplet of T is rooted at $r(G)$. Thus if three genes a, b, c of $L(G)$ form a speciation triplet $s(a)s(b)|s(c) \in T$, then a and b are in some part P_i while c is in another part P_j . But by the definition of speciation partitions, $lca_S(s(P_i))$ is unrelated to $lca_S(s(P_j))$, implying that $s(a)s(b)|s(c) \in tr(S)$. It follows that S displays T .

\Rightarrow : suppose both conditions are not met, but that R is consistent with S . If $R(\emptyset)$ is disconnected but has an inconsistent component, then R is inconsistent by Lemma 4. So we assume $R(\emptyset)$ is connected. If $\overline{R(U)}$ is also connected, then we saw in Theorem 3 that R is not even satisfiable. If $\overline{R(U)}$ is disconnected and admits a speciation partition, but a member of this partition is not consistent, then again by Lemma 4, R is not consistent. So we assume that $R(\emptyset)$ is connected, and $\overline{R(U)}$ is disconnected but admits no speciation partition. Let G be a DS -tree for R consistent with S . Suppose $r(G)$ is a duplication node and let r_1, r_2 be two children of $r(G)$. We have that every gene in $L(G_{r_1})$ is paralogous with every gene in $L(G_{r_2})$ and vice-versa. This implies that $L(G_{r_1})$ and $L(G_{r_2})$ are two components of $R(\emptyset)$ that share no edge, a contradiction since we assume $R(\emptyset)$ is connected. So $r(G)$ is a speciation node. Let r_1, \dots, r_k be the children of $r(G)$. The sets $L(G_{r_1}), \dots, L(G_{r_k})$ form a partition P of the connected components of $\overline{R(U)}$. Since S

displays $tr_S(G)$, it follows that for two distinct $P_i, P_j \in P$, $lca_S(s(P_i))$ and $lca_S(s(P_j))$ are unrelated.

Hence P is a speciation partition, a contradiction.

This theorem suggests a small modification to algorithm BuildDSTree. Connected components of $R(\emptyset)$ are handled in the same manner, but in the case of a disconnected $\overline{R(U)}$, we need to find a speciation partition P after having found its connected components CC . To accomplish this, it suffices to observe that some $C_1, C_2 \in CC$ must be in the same part of P when $lca_S(s(C_1))$ is on the path from $lca_S(s(C_2))$ to the root of S (or vice-versa). Thus for each $P_i \in P$, we can find the member of $C \in P_i$ that has $lca_S(s(C))$ the closest to the root of S , then any other component C' having $lca_S(s(C'))$ in the subtree rooted at $lca_S(s(C))$ will be in P_i .

FINDSPECIATIONPARTITION uses that fact to find P through a pre-order traversal of S .

Algorithm FindSpeciationPartition(CC, s, P, P_i)

where CC is the set of components to partition, $s \in V(S)$ is the current node of S in the pre-order traversal, P is the partition of CC , and P_i is the current part of P we are adding components to

```

FOR  $C \in CC$  such that  $lca_S(C) = s$ ;
  IF  $P_i$  is not set; let  $P_i$  be a new empty set and
  add  $P_i$  to  $P$ 
  Add  $C$  to  $P_i$ 
END FOR
FOR  $s' \in \text{children}(s)$ ;
  FindSpeciationPartition( $CC, s', P, P_i$ )
END FOR

```

Assuming constant time for each lca lookup, we can precompute $lca_S(s(C))$ in time $|C|$ for each $C \in CC$. If CC has a total of k nodes, by mapping each $s \in S$ to the list of $C \in CC$ with $lca_S(s(C)) = s$, the whole algorithm takes time $O(k + |S|)$. We need up to $n - 1$ calls of BuildDSTree. We argued that one call on a node v of G in BUILD DSTREE takes time $O(|L(G_v)|^2)$, so adding this step makes it $O(|L(G_v)|^2 + |S| + k)$. Noting that $k = |L(G_v)|$, and assuming that $|L(G_v)| \geq |S|$, this modified algorithm still runs in time $O(n^3)$, where $n = |G|$.

Consistency of a satisfiable constraint graph

Now let $R = (V, E, U)$ be a constraint graph for \mathcal{G} and suppose the species tree for Σ is unknown. The question is to know whether the graph R is consistent, and if so to output a species tree S such that R is consistent with S . As above, we assume that R is satisfiable. Note that unlike the two previous problems, we cannot treat each connected component of $R(\emptyset)$ or $\overline{R(U)}$ independently, as two (or more) components might give gene histories consistent by themselves but not together.

We hereafter begin with two classes of constraint graphs for which consistency always holds.

Orthology constraints only: Suppose $R = (V = \mathcal{G}, E, U)$ represents a constraint set restricted to orthology constraints, i.e. $C = (C_O, C_P)$, where $C_P = \tilde{C}$ is the trivial set of paralogy constraints. For each $s_i \in \Sigma$, let $L_{s_i} = \{x \in V : s(x) = s_i\}$ and let F_{s_i} be the star-tree joining all the leaves of L_{s_i} under a single duplication node. Now let G be the star-tree joining all the F_{s_i} trees, for all $s_i \in \Sigma$, under a single speciation node. Then G satisfies the full realization $R(U)$ of R and it is consistent with the star-tree for Σ . Therefore any set of orthology constraints is consistent.

Paralogy constraints only: Suppose $R = (V = \mathcal{G}, E, U)$ represents a constraint set restricted to paralogy constraints, i.e. $C = (C_O = \emptyset, C_P)$. If G is the star-tree joining all the genes of \mathcal{G} under a single duplication node, then G satisfies the empty realization $R(\emptyset)$ of R and it is consistent with the star-tree for Σ . Therefore any set of paralogy constraints is consistent.

Consider now a full constraint graph R . The results in [17,18] suggest a polynomial-time algorithm for solving the consistency problem that consists in building a DS -tree G satisfying R , extracting all speciation triplets of G and checking their consistency with a species tree. Here we propose an alternative polynomial time algorithm for the same problem, avoiding the first step of a DS -tree construction. We first introduce the following subset P_3

(R) of triplets of $\binom{V}{3}$ inducing a path of size 3 in R :

$$P_3(R) = \{s(x)s(y)|s(z) : \{x, y, z\} \in \binom{V}{3}, zx, zy \in E \text{ and } xy \notin E \cup U \text{ and } s(x) \neq s(y)\}$$

Notice that $s(x)s(y)|s(z) \in P_3(R)$ implies that any DS -tree G satisfying R has $s(x)s(y)|s(z) \in tr_S(G)$. Indeed, since $xy \notin E \cup U$, $lca_G(x, y)$ is a duplication node. And since both x and y are related to z by speciation, $lca_G(x, z) = lca_G(y, z)$ and $xy|z$ must be a speciation triplet of G .

For example, consider the vertices b_1, c_2, e_1 of the R graph in Figure 1, which form a path of length 3 with e_1 in the center. In the DS -tree G_1 , $lca_{G_1}(b_1, c_2)$ is a duplication, and $lca_{G_1}(b_1, c_2) (\{b_1, c_2, e_1\})$ is a speciation. Restricting G_1 to the three vertices yields the triplet $b_1c_2|e_1$ rooted at a speciation, and therefore, $s(b_1)s(c_2)|s(e_1) \in tr_S(G_1)$.

The same holds for the $s(c_1)s(c_2)|s(e_1)$ triplet implied by the P_3 induced by c_1, c_2, e_1 . Notice however that in both DS -trees, $s(b_1)s(c_1)|s(e_1)$ is a speciation triplet, though b_1, c_1, e_1 do not induce a P_3 . We show that this kind of speciation triplet is implied by the other two aforementioned P_3 , and that the P_3 subgraphs actually imply every mandatory speciation triplet.

Theorem 5 Let $R = (V, E, U = \emptyset)$ be a satisfiable full constraint graph. Then R is consistent if and only if there exists a species tree S displaying all the triplets of $P_3(R)$.

Proof: \Rightarrow : since $s(x)s(y)|s(z) \in P_3(R)$ implies that $s(x)s(y)|s(z) \in tr_S(G)$, it follows that any species tree S consistent with R must display every triplet of $P_3(R)$.

\Leftarrow : we first obtain a least-resolved DS -tree G for R in terms of speciation. Let G' be a consistent DS -tree satisfying R , and let S be a species tree displaying $P_3(R)$. If G' has any speciation node v that has a speciation child w , we obtain G'' by contracting v and w (delete w and give its children to v). Since v and w are both speciations, this operation does not change the label of $lca_G(x, y)$ for any two leaves x and y and G'' still satisfies R . Moreover, $tr_S(G'') \subset tr_S(G')$, so there is no risk of breaking consistency. We obtain a DS -tree G by repeating this operation until we cannot find such a v and w .

Let $xy|z$ be a triplet of G rooted at a speciation node. We have that $lca_G(z, x) = lca_G(z, y)$ is a speciation, and that $zx, zy \in E$. If $lca_G(x, y)$ is a duplication node, then $xy \notin E$. So $\{x, y, z\}$ induces a P_3 in R , and S displays $s(x)s(y)|s(z)$. Suppose instead that $lca_G(x, y)$ is a speciation node. Because G is a least resolved DS -tree, there must be a duplication node u on the path between $lca_G(x, y)$ and $lca_G(x, z)$. This implies there is a leaf d in G_u such that x and y are related to d by duplication, but with d and z related by speciation. In R , we then have $zd \in E$, and $xd, yd \notin E$. Thus both $\{x, d, z\}$ and $\{y, d, z\}$ induce a P_3 in R with z being the middle vertex, and $s(x)s(d)|s(z), s(y)s(d)|s(z) \in P_3(R)$ are both displayed by S . This is only possible if there is a node in S that has all of $s(x), s(y), s(d)$ in one child subtree and $s(z)$ in another. Therefore, S must display $s(x)s(y)|s(z)$. Having taken care of both types of speciation triplets, we deduce that displaying $P_3(R)$ is sufficient to display $tr_S(G)$.

Therefore the consistency problem for a satisfiable full constraint graph reduces to the problem of verifying whether the set $P_3(R)$ of triples can be displayed in a species tree for Σ . This is in fact a well know problem with a solution presented in [22]: given a triplet set \mathcal{R} , there is a polynomial-time algorithm, called BUILD [23], that, when applied to \mathcal{R} either outputs a species tree that displays \mathcal{R} or recognizes that \mathcal{R} is inconsistent. Therefore, in the case of a full constraint graph, the consistency problem is resolved in polynomial time by first constructing the set $P_3(R)$, and then applying the BUILD algorithm.

Consider now the general case of a constraint graph $R = (V, E, U)$ with $U \neq \emptyset$. The branch-and-bound algorithm CHECKCONS iterates over the edges of U , tries to make them edges and non-edges but stops as soon as one decision creates a set of P_4 with no unknowns or a set of P_3 that is inconsistent. Since at worst, every possibility is tested, it follows that this algorithm is exact, though exponential in the worst case.

ALGORITHM CHECKCONS ($R = (V, E, U)$)

```

Obtain a species tree  $S$  by running BUILD on  $P_3(R)$ 
IF  $S$  is not set (i.e. BUILD failed), RETURN FALSE
IF  $R$  is not satisfiable, RETURN FALSE
IF  $U = \emptyset$ , return ( $R, S$ )
Let  $e \in U$  and let  $R_e = (V, E \cup \{e\}, U \setminus \{e\})$ 
( $R', S$ )  $\leftarrow$  CHECKCON  $S(R_e)$ 
IF ( $R', S$ ) is set (i.e. CHECKCONS succeeded),
RETURN ( $R', S$ ) Otherwise let  $R_{\bar{e}} = (V, E, U \setminus \{e\})$ 
( $R', S$ )  $\leftarrow$  CHECKCONS( $R_{\bar{e}}$ )
IF ( $R', S$ ) is set (i.e. CHECKCONS succeeded),
RETURN ( $R', S$ )
Otherwise RETURN FALSE
    
```

Possible improvements of this algorithm include removing as many edges from U as possible, and choosing an ordering of the edges that may speed up the branch-and-bound process. For instance, it may be worthwhile to first identify every induced P_4 of $R(\emptyset)$. The P_4 subgraphs that admit only one possibility for removal, i.e. the P_4 can only be removed by making a unique edge $e \in U$ an orthology edge, can be corrected before entering the algorithm. Note that the same applies for the edges of U that must be edges of $\overline{R(U)}$. We may then prioritize the handling of the other P_4 by considering the edges that resolve them first. Similarly, it would also be possible to identify edges of U that are mandatory in E by finding the P_3 subgraphs of $R(\emptyset)$ that are not in $P_3(R)$, but that disagree with a triplet of $P_3(R)$. For instance, $R(\emptyset)$ might have a P_3 with edges xz, zy , but this P_3 is not in $P_3(R)$ because $xy \in U$. If say $s(y)s(z)|s(x)$ is in $P_3(R)$, then xy is forced in E as otherwise the contradictory $s(x)s(y)|s(z)$ triplet would be present.

Experiments

We show how the developed algorithms for checking satisfiability and consistency can be used, in combination with an orthology detection tool such as ProteinOrtho [11], to infer a robust set of orthology and paralogy constraints. Given a set of protein sequences, Proteinortho infers homologous gene families as well as orthology relationships within these families, based on various similarity scores. Proteinortho does not infer paralogy relationships. However, if we choose a set of parameters leading to a loose characterization of orthologs, then we can assume that unpredicted constraints should represent paralogy. Different combinations of parameters therefore lead to different constraint sets that can be analyzed for satisfiability and consistency.

ProteinOrtho has been run on 265 gene families of vertebrates, each representing the leaf-set of an *Ensembl* [5] gene tree. Trees were chosen randomly among the *Ensembl* gene trees containing at least 20 leaves. For

each family, five different parameter settings, numbered from -2 to $+2$, were tested, 0 representing the default parameter choice of ProteinOrtho, and the smaller the number, the looser is the induced characterization of orthology. For each parameter setting i , we define the full constraint graph R^i where all gene pairs not predicted as orthologs are interpreted as paralogs. Typically, a graph R^- for a negative number (-1 or -2) contains more orthology (and thus less paralogy) constraints than R^0 , while the converse is true for a graph R^+ . Combining two constraint graphs R^- and R^+ consists in keeping only orthology and paralogy edges that are common to both, and completing the graph with unknown edges.

Table 1 summarizes the results on satisfiability and consistency with the *Ensembl* species tree S , obtained for each gene family and each parameter setting or combination. Among the 265 gene families, only 112 (42%) produced at least one satisfiable full constraint graph and only 44 (15%) produced such a graph which is also consistent with the *Ensembl* species tree. However, combining loose and strict parameter settings lead to much better results with at least 95% satisfiability and 56% consistency with S . The partial orthology/paralogy constraints obtained from combinations correspond to about half of the constraints of a full graph, as illustrated by the last column of the table.

In order to get a rough idea of the accuracy of the obtained partial orthology/paralogy predictions for each gene family \mathcal{G} , we compared them with those resulting from the labeling of the *Ensembl* gene tree nodes as duplication and speciation nodes. An *orthology disagreement* refers to orthology predictions on the four combined graphs depicted in Table 1, that are rather inferred as paralogs from the *Ensembl* gene tree labeling.

Table 1 The results over 265 gene families from Ensembl.

	# satisfiable families	# consistent families	% constraints when consistent
-2	82 (30.9%)	30 (11.3%)	
-1	44 (16.6%)	13 (4.91%)	
0	26 (9.81%)	9 (3.40%)	
+1	48 (18.1%)	14 (5.28%)	
+2	55 (20.8%)	18 (6.79%)	
-2/+2	260 (98.1%)	172 (64.9%)	42.0%
-2/+1	258 (97.4%)	172 (64.9%)	44.8%
-1/+1	254 (95.8%)	149 (56.2%)	50.6%
-1/+2	255 (95.9%)	157 (59.2%)	47.5%

The first five rows correspond to the full constraint sets obtained from PROTEINORTHO for the five classes of parameters. The last four rows correspond to the partial constraint sets obtained after combining two graphs over two types of parameters. The first column is the number of families for which the settings of PROTEINORTHO yielded a satisfiable graph, the second that number consistent with the *Ensembl* species tree. The last column shows the percentage of constraints that were not unknown when a consistent solution was found (which is 100% for the first five rows).

A *paralogy disagreement* refers to the reverse situation. Overall, the orthology disagreement percentage is between 15.1% and 15.9% depending on the two classes of parameters combined. For paralogy disagreement, it varies between 11% and 17%, depending on the 2 parameters combined ($-2/+1$ and $-2/+2$ were around 11.2% while $-1/+1$ and $-1/+2$ were around 17.4%).

Notice that *Ensembl* annotates many duplication nodes as “dubious”. If we ignore orthology disagreements caused by a dubious duplication node, the orthology disagreement percentage drops to an average of 5.0%, strengthening the doubts on those duplication nodes.

Conclusion

In this work we have developed methods to assess the plausibility of a partial set of orthology and paralogy relationships between pairs of homologous genes. In particular, we showed how extending algorithms for the *Graph sandwich problem* can solve, in cubic time, the problems of satisfiability and consistency with a given species tree. In case of an unknown species tree, the complexity of the problem of verifying consistency with *some* species tree remains open. We have elaborated on the P_3 property of the constraint graph, which leads to an exponential branch-and-bound algorithm. It remains possible that this property could be used to create a more efficient method. While previous work consisted in verifying whether a full set of relationships was satisfiable or consistent, admitting uncertainty within these relationships makes it possible to bring the theory from [18] into practice, as current orthology (or paralogy) inference methods based on sequences cannot guarantee 100% accuracy in their predictions. We show how a confidence set of such predictions can be inferred using our methods and Proteinortho. A promising direction is to use such robust predictions to correct gene trees in case of disagreement.

Availability

Software is available at <http://www-ens.iro.umontreal.ca/~lafonman/software.php>.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ML, NE devised the proofs and algorithms and wrote the paper. ML implemented the software.

Declarations

Publication charges for this work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant and Fonds de Recherche Nature et Technologies of Quebec (FQRNT) B2 Grant. This article has been published as part of *BMC Genomics* Volume 15 Supplement 6, 2014: Proceedings of the Twelfth Annual Research in

Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics. The full contents of the supplement are available online at <http://www.biomedcentral.com/bmcgenomics/supplements/15/S6>.

Published: 17 October 2014

References

1. Ohno S: *Evolution by gene duplication* Berlin: Springer; 1970.
2. Goodman M, Czelusniak J, Moore G, Romero-Herrera A, Matsuda G: **Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences.** *Systematic Zoology* 1979, **28**:132-163.
3. Coghlan A, Coin JRL, Heriche J, Osmotherly L, Li R, T TL, Zhang Z, Bolund L, Wong G, Zheng W, Dehal P, J JW, Durbin R: **TreeFam: a curated database of phylogenetic trees of animal gene families.** *Nucleic Acids Research* 2006, **34**:D572-580.
4. Schreiber F, Patricio M, Muffato M, Pignatelli M, Bateman A: **TreeFam v9: a new website, more species and orthology-on-the-fly.** *Nucleic Acids Res* 2013, **42**:D922-D925.
5. Flicek P, Amode M, Barrell D, Beal K, Brent S, Carvalho-Silva D, Clapham P, Coates G, Fairley S, Fitzgerald S, Gil L, Gordon L, Hendrix M, Hourlier T, Johnson N, Kahari A, Keefe D, Keenan S, Kinsella R, Komorowska M, Koscielny G, Kulesha E, Larsson P, Longden I, McLaren W, Muffato M, Overduin B, Pignatelli M, Pritchard B, Riati H, Ritchie G, Ruffier M, Schuster M, Sobral D, Tang Y, Taylor K, Trevanion S, Vandrovцова J, White S, Wilson M, Wilder S, Aken B, Birney E, Cunningham F, Dunham I, Durbin R, Fernandez J, Infante J, Jones A, Jones C, Jones P, Jones R, Jones S, Jones T, Jones V, Jones W, Jones X, Jones Y, Jones Z, Jones AA, Jones AB, Jones AC, Jones AD, Jones AE, Jones AF, Jones AG, Jones AH, Jones AI, Jones AJ, Jones AK, Jones AL, Jones AM, Jones AN, Jones AO, Jones AP, Jones AQ, Jones AR, Jones AS, Jones AT, Jones AU, Jones AV, Jones AW, Jones AX, Jones AY, Jones AZ, Jones BA, Jones BB, Jones BC, Jones BD, Jones BE, Jones BF, Jones BG, Jones BH, Jones BI, Jones BJ, Jones BK, Jones BL, Jones BM, Jones BN, Jones BO, Jones BP, Jones BQ, Jones BR, Jones BS, Jones BT, Jones BU, Jones BV, Jones BW, Jones BX, Jones BY, Jones BZ, Jones CA, Jones CB, Jones CC, Jones CD, Jones CE, Jones CF, Jones CG, Jones CH, Jones CI, Jones CJ, Jones CK, Jones CL, Jones CM, Jones CN, Jones CO, Jones CP, Jones CQ, Jones CR, Jones CS, Jones CT, Jones CU, Jones CV, Jones CW, Jones CX, Jones CY, Jones CZ, Jones DA, Jones DB, Jones DC, Jones DD, Jones DE, Jones DF, Jones DG, Jones DH, Jones DI, Jones DJ, Jones DK, Jones DL, Jones DM, Jones DN, Jones DO, Jones DP, Jones DQ, Jones DR, Jones DS, Jones DT, Jones DU, Jones DV, Jones DW, Jones DX, Jones DY, Jones DZ, Jones EA, Jones EB, Jones EC, Jones ED, Jones EE, Jones EF, Jones EG, Jones EH, Jones EI, Jones EJ, Jones EK, Jones EL, Jones EM, Jones EN, Jones EO, Jones EP, Jones EQ, Jones ER, Jones ES, Jones ET, Jones EU, Jones EV, Jones EW, Jones EX, Jones EY, Jones EZ, Jones FA, Jones FB, Jones FC, Jones FD, Jones FE, Jones FF, Jones FG, Jones FH, Jones FI, Jones FJ, Jones FK, Jones FL, Jones FM, Jones FN, Jones FO, Jones FP, Jones FQ, Jones FR, Jones FS, Jones FT, Jones FU, Jones FV, Jones FW, Jones FX, Jones FY, Jones FZ, Jones GA, Jones GB, Jones GC, Jones GD, Jones GE, Jones GF, Jones GG, Jones GH, Jones GI, Jones GJ, Jones GK, Jones GL, Jones GM, Jones GN, Jones GO, Jones GP, Jones GQ, Jones GR, Jones GS, Jones GT, Jones GU, Jones GV, Jones GW, Jones GX, Jones GY, Jones GZ, Jones HA, Jones HB, Jones HC, Jones HD, Jones HE, Jones HF, Jones HG, Jones HI, Jones HJ, Jones HK, Jones HL, Jones HM, Jones HN, Jones HO, Jones HP, Jones HQ, Jones HR, Jones HS, Jones HT, Jones HU, Jones HV, Jones HW, Jones HX, Jones HY, Jones HZ, Jones IA, Jones IB, Jones IC, Jones ID, Jones IE, Jones IF, Jones IG, Jones IH, Jones II, Jones IJ, Jones IK, Jones IL, Jones IM, Jones IN, Jones IO, Jones IP, Jones IQ, Jones IR, Jones IS, Jones IT, Jones IU, Jones IV, Jones IW, Jones IX, Jones IY, Jones IZ, Jones JA, Jones JB, Jones JC, Jones JD, Jones JE, Jones JF, Jones JG, Jones JH, Jones JI, Jones JJ, Jones JK, Jones JL, Jones JM, Jones JN, Jones JO, Jones JP, Jones JQ, Jones JR, Jones JS, Jones JT, Jones JU, Jones JV, Jones JW, Jones JX, Jones JY, Jones JZ, Jones KA, Jones KB, Jones KC, Jones KD, Jones KE, Jones KF, Jones KG, Jones KH, Jones KI, Jones KJ, Jones KL, Jones KM, Jones KN, Jones KO, Jones KP, Jones KQ, Jones KR, Jones KS, Jones KT, Jones KU, Jones KV, Jones KW, Jones KX, Jones KY, Jones KZ, Jones LA, Jones LB, Jones LC, Jones LD, Jones LE, Jones LF, Jones LG, Jones LH, Jones LI, Jones LJ, Jones LK, Jones LL, Jones LM, Jones LN, Jones LO, Jones LP, Jones LQ, Jones LR, Jones LS, Jones LT, Jones LU, Jones LV, Jones LW, Jones LX, Jones LY, Jones LZ, Jones MA, Jones MB, Jones MC, Jones MD, Jones ME, Jones MF, Jones MG, Jones MH, Jones MI, Jones MJ, Jones MK, Jones ML, Jones MN, Jones MO, Jones MP, Jones MQ, Jones MR, Jones MS, Jones MT, Jones MU, Jones MV, Jones MW, Jones MX, Jones MY, Jones MZ, Jones NA, Jones NB, Jones NC, Jones ND, Jones NE, Jones NF, Jones NG, Jones NH, Jones NI, Jones NJ, Jones NK, Jones NL, Jones NM, Jones NN, Jones NO, Jones NP, Jones NQ, Jones NR, Jones NS, Jones NT, Jones NU, Jones NV, Jones NW, Jones NX, Jones NY, Jones NZ, Jones OA, Jones OB, Jones OC, Jones OD, Jones OE, Jones OF, Jones OG, Jones OH, Jones OI, Jones OJ, Jones OK, Jones OL, Jones OM, Jones ON, Jones OO, Jones OP, Jones OQ, Jones OR, Jones OS, Jones OT, Jones OU, Jones OV, Jones OW, Jones OX, Jones OY, Jones OZ, Jones PA, Jones PB, Jones PC, Jones PD, Jones PE, Jones PF, Jones PG, Jones PH, Jones PI, Jones PJ, Jones PK, Jones PL, Jones PM, Jones PN, Jones PO, Jones PP, Jones PQ, Jones PR, Jones PS, Jones PT, Jones PU, Jones PV, Jones PW, Jones PX, Jones PY, Jones PZ, Jones QA, Jones QB, Jones QC, Jones QD, Jones QE, Jones QF, Jones QG, Jones QH, Jones QI, Jones QJ, Jones QK, Jones QL, Jones QM, Jones QN, Jones QO, Jones QP, Jones QQ, Jones QR, Jones QS, Jones QT, Jones QU, Jones QV, Jones QW, Jones QX, Jones QY, Jones QZ, Jones RA, Jones RB, Jones RC, Jones RD, Jones RE, Jones RF, Jones RG, Jones RH, Jones RI, Jones RJ, Jones RK, Jones RL, Jones RM, Jones RN, Jones RO, Jones RP, Jones RQ, Jones RR, Jones RS, Jones RT, Jones RU, Jones RV, Jones RW, Jones RX, Jones RY, Jones RZ, Jones SA, Jones SB, Jones SC, Jones SD, Jones SE, Jones SF, Jones SG, Jones SH, Jones SI, Jones SJ, Jones SK, Jones SL, Jones SM, Jones SN, Jones SO, Jones SP, Jones SQ, Jones SR, Jones SS, Jones ST, Jones SU, Jones SV, Jones SW, Jones SX, Jones SY, Jones SZ, Jones TA, Jones TB, Jones TC, Jones TD, Jones TE, Jones TF, Jones TG, Jones TH, Jones TI, Jones TJ, Jones TK, Jones TL, Jones TM, Jones TN, Jones TO, Jones TP, Jones TQ, Jones TR, Jones TS, Jones TT, Jones TU, Jones TV, Jones TW, Jones TX, Jones TY, Jones TZ, Jones UA, Jones UB, Jones UC, Jones UD, Jones UE, Jones UF, Jones UG, Jones UH, Jones UI, Jones UJ, Jones UK, Jones UL, Jones UM, Jones UN, Jones UO, Jones UP, Jones UQ, Jones UR, Jones US, Jones UT, Jones UU, Jones UV, Jones UW, Jones UX, Jones UY, Jones UZ, Jones VA, Jones VB, Jones VC, Jones VD, Jones VE, Jones VF, Jones VG, Jones VH, Jones VI, Jones VJ, Jones VK, Jones VL, Jones VM, Jones VN, Jones VO, Jones VP, Jones VQ, Jones VR, Jones VS, Jones VT, Jones VU, Jones VV, Jones VW, Jones VX, Jones VY, Jones VZ, Jones WA, Jones WB, Jones WC, Jones WD, Jones WE, Jones WF, Jones WG, Jones WH, Jones WI, Jones WJ, Jones WK, Jones WL, Jones WM, Jones WN, Jones WO, Jones WP, Jones WQ, Jones WR, Jones WS, Jones WT, Jones WU, Jones WV, Jones WW, Jones WX, Jones WY, Jones WZ, Jones XA, Jones XB, Jones XC, Jones XD, Jones XE, Jones XF, Jones XG, Jones XH, Jones XI, Jones XJ, Jones XK, Jones XL, Jones XM, Jones XN, Jones XO, Jones XP, Jones XQ, Jones XR, Jones XS, Jones XT, Jones XU, Jones XV, Jones XW, Jones XX, Jones XY, Jones XZ, Jones YA, Jones YB, Jones YC, Jones YD, Jones YE, Jones YF, Jones YG, Jones YH, Jones YI, Jones YJ, Jones YK, Jones YL, Jones YM, Jones YN, Jones YO, Jones YP, Jones YQ, Jones YR, Jones YS, Jones YT, Jones YU, Jones YV, Jones YW, Jones YX, Jones YY, Jones YZ, Jones ZA, Jones ZB, Jones ZC, Jones ZD, Jones ZE, Jones ZF, Jones ZG, Jones ZH, Jones ZI, Jones ZJ, Jones ZK, Jones ZL, Jones ZM, Jones ZN, Jones ZO, Jones ZP, Jones ZQ, Jones ZR, Jones ZS, Jones ZT, Jones ZU, Jones ZV, Jones ZW, Jones ZX, Jones ZY, Jones ZZ.

22. Semple C, Steel M: **Phylogenetics**. In *Oxford Lecture Series in Mathematics and in Applications. Volume 24*. [Oxford, UK: Oxford University Press]; 2003:119-120.
23. Aho A, Sagiv Y, Szymanski T, Ullman J: **Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions**. *SIAM J Comput* 1981, **10**:405-421.

doi:10.1186/1471-2164-15-S6-S12

Cite this article as: Lafond and El-Mabrouk: **Orthology and paralogy constraints: satisfiability and consistency**. *BMC Genomics* 2014 **15**(Suppl 6):S12.

**Submit your next manuscript to BioMed Central
and take full advantage of:**

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at
www.biomedcentral.com/submit

