BIOINFORMATICS ORIGINAL PAPER Vol. 21 no. 11 2005, pages 2596–2603
doi:10.1093/bioinformatics/bti325

Sequence analysis

Tree pattern matching in phylogenetic trees: automatic search for orthologs or paralogs in homologous gene sequence databases

Jean-François Dufayard[1], Laurent Duret[2], Simon Penel[2], Manolo Gouy[2], François Rechenmann[1] and Guy Perrière[2,*]

[1]INRIA Rhône-Alpes, 38334 Montbonnot, Saint Ismier Cedex, France and [2]Laboratoire de Biométrie et Biologie Évolutive, UMR CNRS 5558, Université Claude Bernard—Lyon 1, 43 bd. du 11 Novembre 1918, 69622 Villeurbanne Cedex, France

Received on May 7, 2004; revised on January 13, 2005; accepted on February 9, 2005
Advance Access publication February 15, 2005

ABSTRACT

Motivation: Comparative sequence analysis is widely used to study genome function and evolution. This approach first requires the identification of homologous genes and then the interpretation of their homology relationships (orthology or paralogy). To provide help in this complex task, we developed three databases of homologous genes containing sequences, multiple alignments and phylogenetic trees: HOBACGEN, HOVERGEN and HOGENOM. In this paper, we present two new tools for automating the search for orthologs or paralogs in these databases.

Results: First, we have developed and implemented an algorithm to infer speciation and duplication events by comparison of gene and species trees (tree reconciliation). Second, we have developed a general method to search in our databases the gene families for which the tree topology matches a peculiar tree pattern. This algorithm of unordered tree pattern matching has been implemented in the FamFetch graphical interface. With the help of a graphical editor, the user can specify the topology of the tree pattern, and set constraints on its nodes and leaves. Then, this pattern is compared with all the phylogenetic trees of the database, to retrieve the families in which one or several occurrences of this pattern are found. By specifying *ad hoc* patterns, it is therefore possible to identify orthologs in our databases.

Availability: The tree reconciliation program and the FamFetch interface are available from the Pôle Bioinformatique Lyonnais Web server at the following addresses: http://pbil.univ-lyon1.fr/software/RAP/RAP.htm and http://pbil.univ-lyon1.fr/software/famfetch.html

Contact: perriere@biomserv.univ-lyon1.fr

INTRODUCTION

Comparison of homologous sequences is an essential step for many studies related to molecular biology and evolution. For instance, it is used in the prediction of gene function, protein or RNA structure prediction, study of genome duplications, comparative mapping or molecular phylogeny. The importance of comparative genomics for deciphering the genetic information embedded within genomes is now widely recognized and, hence, large scale sequencing projects have been set up, resulting in the identification of hundreds of thousands of genes. In this context, we developed three databases gathering genes into homologous families: HOVERGEN (Duret *et al*., 1999) devoted to vertebrates, HOBACGEN (Perrière *et al*., 2000) devoted to prokaryotes, and HOGENOM, devoted to completely sequenced organisms. In these databases, homologous protein genes are classified into families on the basis of BLAST (Altschul *et al*., 1997) similarity searches between protein sequences and, for each family, a multiple alignment and a phylogenetic tree are computed (see Perrière *et al*., 2000, for a complete description of the procedure). These databases are very large, and they contain thousands of families and associated trees. For example, there are 9926 families containing at least 3 genes in the release 46 of HOVERGEN (June 2004).

The interpretation of homology relationships in such large datasets is a complex task. Notably, among homologous genes, one has to distinguish orthologous from paralogous sequences. Orthologs are homologous genes in different species that diverged from a single ancestral gene after a speciation event and paralogs are homologous genes that originate from the intragenomic duplication of an ancestral gene. This distinction is important to predict the function of a new gene by homology, because gene duplications are often followed by changes of function, in one or both of the paralogs (e.g. change in expression pattern, or in the biochemical activity of the encoded protein) (Lynch *et al*., 2001). Hence, orthologous sequences are more reliable predictors of protein function than paralogous sequences. However, changes of function can also occur during the evolution of non-duplicated genes. Thus, although duplications probably lead to an increase in the rate of functional divergence between homologs, closely related paralogs have certainly more similar functions than distantly related orthologs. In other words, to predict the function of a gene by homology, it is necessary to consider not only whether genes are orthologs or paralogs, but also the evolutionary distance between them. It is also important to stress that orthology relationship is not necessarily one-to-one, but can be one-to-many or many-to-many. The distinction between orthologs and paralogs is also useful for other types of analyses such as molecular phylogeny or comparative mapping of different species.

*To whom correspondence should be addressed.

The most rigorous approach in determining whether homologous genes are orthologous or paralogous consists in comparing the gene tree with the species tree considered as a reference. The problem is that this work is very tedious, and automated systems are required for large scale studies (e.g. identify all available orthologous genes between two species for which the complete genome is available). Therefore, we present in this paper two complementary tools that allow the automatic search for orthologs or paralogs within our families databases. First, we propose an algorithm (called RAP) to infer speciation and duplication events, by comparison of gene and species trees (tree reconciliation). Second, we have developed a general method to search gene families for which the tree topology matches a peculiar pattern. A tree pattern is a peculiar tree structure, with various taxonomic and evolutionary parameters contained in nodes and leaves. It can be also considered as a subtree which is a part of a larger tree. These two programs have been implemented under the FamFetch client/server architecture we have developed to query the HOVERGEN, HOBAGEN and HOGENOM databases. With the help of a graphical editor implemented in the FamFetch interface (Perrière *et al*., 2000), the user can specify the topology of the tree pattern, and set constraints on its nodes (duplication or speciation) and leaves (taxa to be included or excluded). Then, this tree pattern is compared with all the phylogenetic trees of the database in order to retrieve the families in which one or several of its occurrences are found. By this way, it is possible to automatically retrieve all orthologs among a given set of species. This system is not limited to the identification of orthologs as it can be used to retrieve any complex tree pattern. For example, it is possible to search for events of gene loss or gene transfer, or to search for gene duplication events.

SYSTEM AND METHODS

Tree reconciliation

The standard procedure to determine whether a node in a phylogenetic tree corresponds to a speciation of a duplication event consists in comparing the gene tree with the species tree. Efficient algorithms have been proposed to solve this problem of tree reconciliation (Page and Charleston, 1997; Eulenstein *et al*., 1998; Ma *et al*., 2000; Zmasek and Eddy, 2001). However, an important limitation to their use is that they require completely resolved (i.e. completely binary) gene and species trees. In fact, species trees often have ambiguities, due to limitations in available paleontological and molecular data. Notably, the taxonomy database at National Center for Biotechnology Information (NCBI) (Wheeler *et al*., 2004), which is used as a reference for the taxonomic classification in sequence databases, contains a large number of unresolved nodes (i.e. multifurcations). On the other hand, gene trees are rarely completely reliable, because of limitations in the number of informative sites in sequence alignments, and because of approximations in the evolutionary models and algorithms that are presently used in molecular phylogeny.

Thus, when the gene tree contradicts the species tree, it is required to assess the reliability of the gene tree. This can be done by bootstrap values, or—when these values are not available—by considering the length of internal branches. This is another limitation of the previous algorithms, as they take into account only the topology of the gene and species trees, but not the length of their branches. To circumvent these problems, we propose an improved algorithm for tree reconciliation, allowing the presence of unresolved nodes both in the gene tree and in the species tree, and taking into account not only the tree topology, but also branch lengths. This algorithm is based on the tree mapping method (Page and Charleston, 1997; Eulenstein *et al*., 1998; Ma *et al*., 2000; Zmasek and Eddy, 2001) that consists in the comparison of a gene tree and its species tree, node by node, using a congruence function. The result of the comparison of the gene tree G and the species tree S is a third tree, the reconciled tree R (Fig. 1). The first step of the method is to define R with the same topology as S. Then, R (equivalent of S) and G are stepped simultaneously: for each incongruent pair of nodes, a duplication node is inserted in R and gene losses are annotated. In the example given in Figure 1, the roots of G and S are the only pair of incongruent nodes.

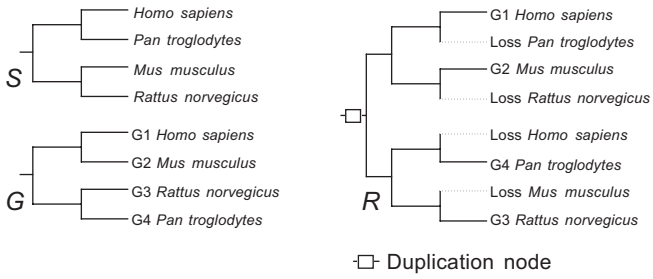

Fig. 1. Tree reconciliation between a gene tree G and a species tree S showing different topologies. The result is the reconciled tree R. R is a variation of S, in which duplication nodes have been inserted in order to explain incongruence with G.

Our algorithm is intended to be used on the homologous gene family databases we developed, and a problem is that these datasets often include redundant sequences. Although efforts are made to minimize redundancy, there are many cases where a single protein is represented by several entries. These redundant sequences are often not exactly identical, either because of polymorphism, sequencing errors, or because they correspond to alternative splice variants. With such data, the standard tree reconciliation algorithms would tend to overestimate the number of gene duplications, because these redundant sequences would be interpreted as paralogs. To solve that problem, we have added a functionality in our algorithm so that two sequences from the same species are considered as paralogs only if they are more divergent than a given threshold, fixed by the user.

Tree pattern matching

The peculiarity of phylogenetic trees—when compared with other trees—is that their leaves are unordered: it means that the trees $((X, Y), Z)$, $((Y, X), Z)$, $(Z, (X, Y))$ and $(Z, (Y, X))$ are all equivalent. It is possible to formulate the unordered tree pattern matching problem as follows:

Let the trees be T and P:

$$T = (V, E, \text{root}(T)), \tag{1}$$

where V is the set of T labeled vertices (nodes) and E the set of T edges (branches).

$$P = (W, F, \text{root}(M)), \tag{2}$$

where W is the set of P labeled vertices and F the set of P edges. P is considered as the tree pattern and T as the target tree. P is a pattern of T if and only if an injective function f can be defined as follows:

$f : \text{w} \in W \rightarrow v \in V$, a node of T is associated to each node of P.

$f(u) = f(v)$ if and only if $u = v$.

label(u)=label($f(u)$).

u is an ancestor of v if and only if $f(u)$ is an ancestor of $f(v)$.

The unordered tree pattern matching problem is well known in computer science (Aho *et al*., 1989; Kilpeläinen and Mannila, 1993), and it has been shown to belong to the NP-complete class (Kilpeläinen, 1992). Moreover, for performing searches, both the target tree and tree pattern have to be rooted. Another problem is the fact that, very often, gene trees are not reliable

and some of their parts may be erroneous. This is due to the limitations of phylogenetic reconstruction methods linked to saturation problems, long branch attraction artefacts or the difficulty of taking into account differences in evolutionary rates. In order to cope with possible errors in the trees, we introduced the use of wildcards in the pattern searches. Such wildcards can be represented by multifurcations. We will therefore consider that the tree pattern (X, Y, Z) matches with the three possible target trees ((X, Y), Z), (X, (Y, Z)) and ((X, Z), Y). (*Note.* Phylogenetic trees are binary trees and hence a true multifurcation cannot exist in a target tree.)

ALGORITHMS

Tree reconciliation

In this section, we describe an implementation of the tree-mapping algorithm, which isolates the congruence function. The notations used are the following: G indicates a node or a leaf of the gene tree, so it corresponds to the whole tree if G is the root. R indicates a node or a leaf of the reconciled tree, so it corresponds to the whole tree if R is the root. *fG* indicates any child of G, if G is not a leaf and *fR* indicates any child of R if R is not a leaf. *Gs* indicates the set of G children, and *Rs* indicates the set of R children. *Card*(X) is the number of elements in the set X and *Species*(Y) is the number of leaves that are not losses under the node Y.

Reconcile: G, R. Transform R (initialized to the species tree) into the reconciled tree of S and G
{ **Invariant:** G and R are congruent }
{ **First recurrence:**
if !*AreCongruent*(G, R) **then**
CreateDuplication(R, G)
Reconcile(G, R) }
{ *Reconcile*(leaf G, leaf R)
label R with the gene of G }
{ *Reconcile*(node G, node R)
foreach *fG* child of G **do**
let *fR* = *MappingChild*(R, *fG*) **in**
if *AreCongruent*(*fG*, *fR*) **then**
Reconcile(*fG*, *fR*)
else
CreateDuplication(*fR*, *fG*)
Reconcile(*fG*, *fR*) }
{ *CreateDuplication*(node R, node G)
duplicate the sub-topology of R, creating a new node
label as losses in the first node of R every species that are not represented in the first node of G
label as losses in the second node of R every species that are not represented in the second node of G }
{*MappingChild*(node R, node or leaf G)
if R is a duplication **then**
if G is the first node of R father **then**
→ the first child of R
else
→ the second child of R
else
→ the child which is not a loss, and which respects the congruence constraints described above }
{ *AreCongruent*(node G, node R)
if (*Card*(*GChildren*) = *Card*(*RChildren*) **and** ∀ *fG*∈*Gs*, ∃ *fR*∈*Rs*
where (*Species*(*fG*) ⊇ *Species*(*fR*) **and** *Species*(*fG*) ⊆ *Species*(*fR*)))
then
→ true
else
→ false }

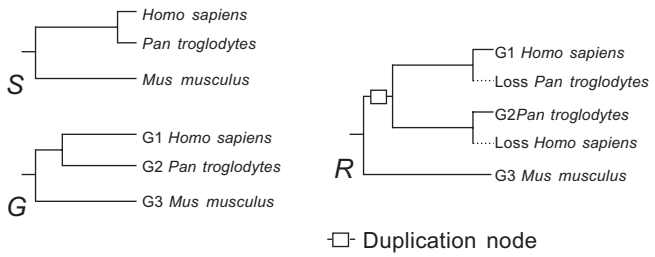

Fig. 2. Tree reconciliation between a gene tree G and a species tree S sharing the same topology but showing differences in branch lengths. Topologies of G and S are identical, but the rate ratio of branch lengths is too high to consider the genes from G as orthologs. A duplication node is created to explain this, and gives the reconciliated tree R.

The tree mapping method uses a very basic definition of congruence; therefore the algorithm listed above is not directly applicable to real data. In the version implemented in RAP, the congruence function is improved in order to deal with n-ary nodes that may be encountered in species trees. Also, many branches in gene trees (and even in species tree) are not absolutely reliable. In this case, the congruence function tries to collapse some branches to make the topology correspond. Branches that are collapsed must be of low reliability, and this is verified considering that the bootstrap value of a given branch is under a threshold score. If the tree is not bootstrapped, branch lengths may be used for the same goal. For that purpose, the congruence function compares branch length ratios of G and S.

The ratio of branch lengths is also used to introduce duplication nodes. In the example given in Figure 2, topologies of G and S are equivalent, but the rate ratio of branch lengths is too high to consider these genes as orthologs. A duplication node is then created to explain such a ratio. The minimum rate ratio before duplication is a parameter of the reconciliation method.

Finally, to deal with polymorphism and redundancy, we consider that two sequences from the same species are considered as paralogs only if they are more divergent than a given threshold. If this is not the case, they are considered as redundant entries in the database.

Tree pattern matching

Here, we describe the tree pattern matching algorithm we have implemented. The notations used are the following: T indicates a leaf or a node of the target tree and P indicates a leaf or a node of the tree pattern. *SearchPattern*(T, P) returns true if P is detected at least one time in T and returns false if it is not detected. *Taxa*(X) returns taxon or taxa labelled on any pattern or target tree node or leaf. *LeftChild*(X) and *RightChild*(X) return the respective children of a pattern or target tree node. *BranchConstraint*(P) returns constraints on the branch just above the node or leaf P. It may contain information like 'no duplication node on this path' or 'no intermediate node on this path'. *Nature*(X) returns speciation or

duplication, depending on the node X. The *SearchPattern* algorithm varies, as explained below, depending on T and P being leaves or internal nodes.

SearchPattern(leaf T, leaf P)
  if *Taxa*(T) is included in *Taxa*(P) **then**
    → true
  else
    → false

The definition of this first case of recurrence allows the use of different taxonomic levels. A pattern leaf P is detected in a target tree leaf if and only if the taxon of T is included in the set of taxa of P. For example, if P is labelled with 'Any mammal but not a rodent' and T with species *Canis familiaris*, the function will return true because the dog is a mammal but not a rodent.

SearchPattern(leaf T, node P)
  → false
SearchPattern(node T, leaf P)
  if *BranchConstraints*(P) are compatible with *Nature*(P) **then**
    → (SearchPattern(LeftChild(T), P) **or** *SearchPattern*
      (*RightChild*(T), P))
  else
    → false

This simple case solves a leaf pattern search in a target tree node. The search is propagated recursively on the whole subtree under T. The only constraints to take care of are the branch constraints. For example, if the branch above P is labelled 'no duplication' and T is a duplication, the search must not be propagated to the T children.

SearchPattern(node T, n-ary node P)
  foreach P-bin, binary version of P
    if *SearchPattern*(T, P-bin) **then**
      → true
    else
      → false

This case solves the problem of a non-binary node P searched in a node T. As explained above, a non-binary node P is detected in target tree node T if and only if at least one binary version of P is detected.

SearchPattern(node T, binary node P)
  let the **boolean** result **in**
  if (*Nature*(P) ≠ *Nature*(T) **and** branch constraints of P are not compatible with *Nature*(T)) **then**
    result = false
  else if (*Nature*(P) ≠ *Nature*(T)) **then**
    result = (*SearchPattern*(*LeftChild*(T), P) **or**
      SearchPattern(*RightChild*(T), P))
  else if branch constraints of P are not compatible with *Nature*(T)
    result = ((*SearchPattern*(*LeftChild*(T), *LeftChild*(P)) **and**
    SearchPattern(*RightChild*(T), *RightChild*(P))) **or**
      (*SearchPattern*(*RightChild*(T),
    LeftChild(P)) **and** *SearchPattern*(*LeftChild*(T),
      RightChild(P)))))
  else
    result = ((*SearchPattern*(*LeftChild*(T), *LeftChild*(P)) **and**
    SearchPattern(*RightChild*(T), *RightChild*(P))) **or**
      (*SearchPattern*(*RightChild*(T),
    LeftChild(P)) **and** *SearchPattern*(*LeftChild*(T),
      RightChild(P))) **or**
    SearchPattern(*LeftChild*(T), P) **or** *SearchPattern*
      (*RightChild*(T), P))
  → *result*

To find the pattern P in tree T, we can have two kinds of hypotheses. First, when comparing a node T with node P, we can suppose that T and P are two matching nodes. Then we must verify that children of P can be found in children of T. These hypotheses will be called as 'matching hypothesis'. Second, if matching hypotheses are wrong, it is possible to propagate P matching to T children: these hypotheses are called 'propagation hypotheses'. But for large phylogenetic trees and tree patterns, the hypotheses and solutions to explore are too numerous and it is not reasonable to apply this simple method to real data. A minor change, though, makes this algorithm efficient on large trees. Indeed, it is frequently useless to consider any hypothesis if the following simple and polynomial verification is done: P is not matching T if Species(P) $\not\subset$ Species(T). For example, it is useless to try to match a human/rat/mouse pattern in a tree (or a tree part) that does not contain any rat gene. This simple verification can be done for each recurrence path of the algorithm.

IMPLEMENTATION

The two algorithms have been implemented under the client/server architecture used for our gene family databases. RAP has been developed in Java 1.4 and it is available from the Pôle Bioinformatique Lyonnais (PBIL) server at http://pbil.univ-lyon1.fr/software/RAP/RAP.htm. All the gene trees of HOVERGEN, HOBACGEN and HOGENOM have been rooted and reconciled with RAP, using as a species tree the phylogeny from the NCBI taxonomy database. We set RAP parameters so as to not overestimate the number of gene duplications: we considered that a node in the gene tree should be interpreted as a speciation event, as far as there is no strong evidence that the gene and species trees are incongruent. Since the trees from the three databases are not bootstrapped, the reliability of each gene tree topology was estimated by taking into account the length of internal branches. As the NCBI tree has no branch lengths, we did not set any value for the rate ratio parameter allowing to infer duplication events when reconciliating a gene tree with the species tree. Also, the threshold for the minimum divergence value below which a group of sequences are considered to be redundant was set to 10%.

Tree pattern matching searches can be composed under the FamFetch interface, which also allows to perform many other kind of queries (based on keywords, sequence names or accession numbers, families accession numbers, or by taxa crossing). FamFetch is also a Java application that can be installed on most operating systems (Windows, Unix/Linux and MacOS). It is available at http://pbil.univ-lyon1.fr/software/famfetch.html. The pattern editor of FamFetch is made of two frames: the tool frame and the pattern frame (Fig. 3). The pattern frame is an interactive editor that permits the construction of any pattern, node by node and leaf by leaf. Patterns can be loaded, saved and matched with a tree database from

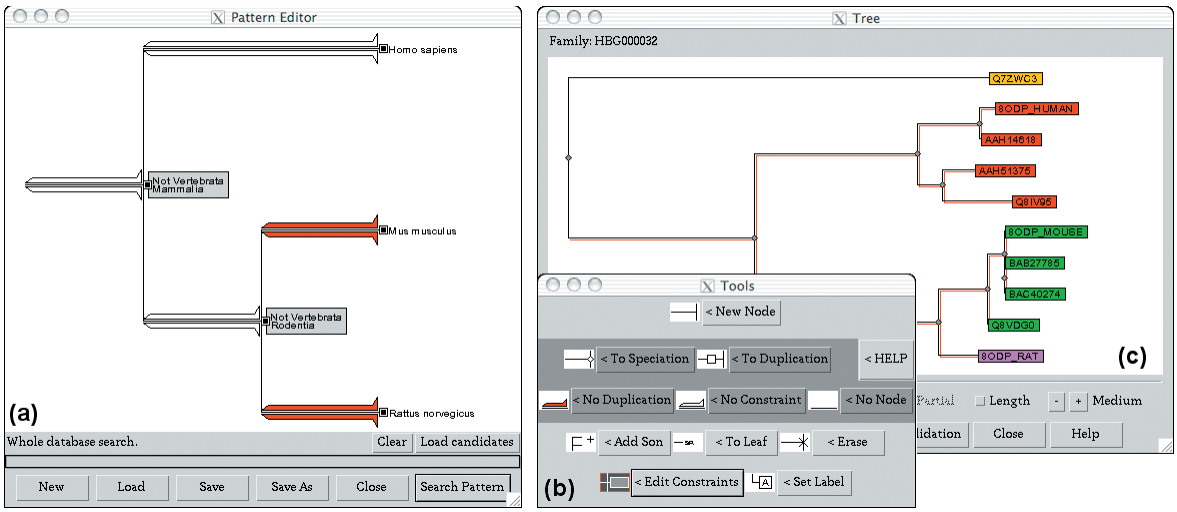

Fig. 3. The two frames of the pattern editor and the tree frame of the FamFetch interface. Frame (**a**) is an interactive editor that permits one to construct any pattern, node by node and leaf by leaf. Frame (**b**) allows to choose between tools to use in the upper frame. Tools surrounded by dark grey are those that use the gene duplication predictions, and can be avoided if the user does not want to trust this information. If the families have been selected by a tree pattern matching operation, retrieved patterns are shown with red lines on each tree in the tree frame (**c**).

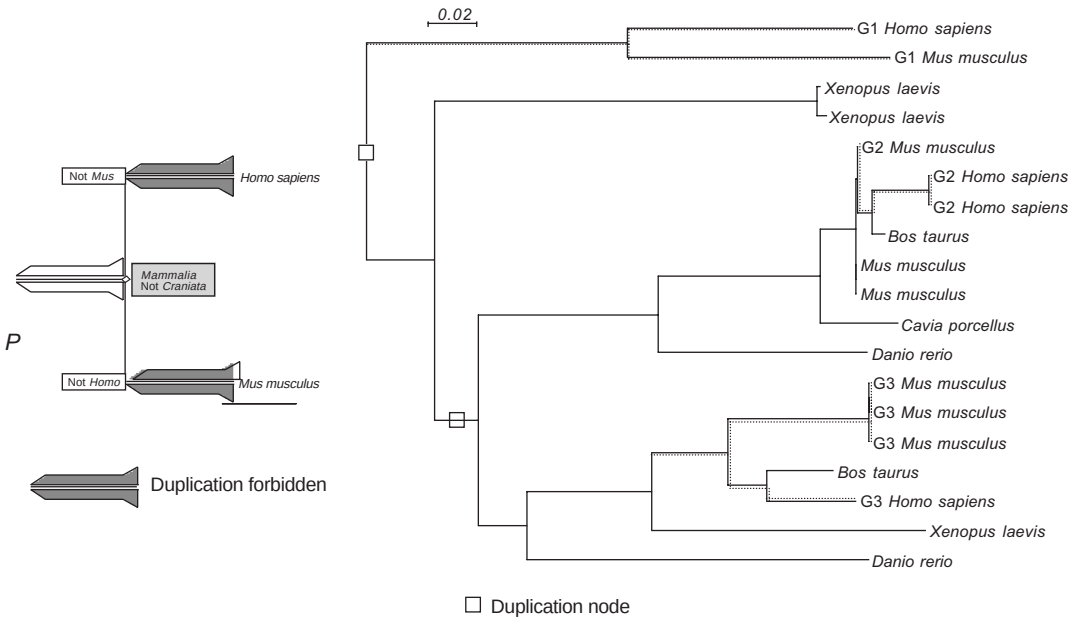

Fig. 4. An example of a query to detect one-to-one orthologous genes in HOVERGEN. In the pattern *P* that has been set, no *Mus musculus* sequences are allowed in the branch leading to *Homo sapiens* and no human sequences are allowed in the branch leading to the mouse. Also, duplications are forbidden in these two branches. The tree *T* displayed on the right corresponds to one of the 9144 families from HOVERGEN that match that query. This family contains sequences of Rho GDP-dissociation inhibitors (OMIM entry 602843) and is highly conserved among taxa. This is a family of interest because it contains three groups of orthologs matching the pattern, instead of a single one. Each group of orthologs is indicated by a dashed line doubling the portions of the tree corresponding to *P*. Note that the second group of orthologs (G2) contains two human sequences while the third group (G3) contains three mouse sequences. But, as these sequences are very similar, so they are considered as redundant.

this frame. The tool frame allows choosing between tools to use in the pattern frame.

The possibilities provided by these tools are: (1) add a new node to any part of the tree pattern; (2) set unresolved topologies (i.e. multifurcations) for some nodes of the pattern; (3) turn a node or a branch into 'speciation only' or 'duplication only'; (4) turn a node to leaf; (5) delete a part of the tree and (6) add taxa constraints to nodes and leaves of the pattern, using any taxonomic level. After the pattern matching operation, the main frame of FamFetch displays the list of matching families. Finally, the results can be saved

in a flat file, each pattern being numbered and described with its gene list.

Thanks to the possibility of introducing duplications and/or taxonomic data constraints in search patterns, it is possible to easily detect ancient gene duplications or to select orthologous genes. This last feature is of special importance as ortholog identification is a key point when establishing molecular phylogenies. For that purpose, the user has only to build a pattern in which duplications are forbidden (Fig. 4). Also, due to the fact that the trees have been reconciliated with RAP, even hidden paralogies due to duplications followed by gene losses in some lineages are taken into account. For instance, a search of all orthologous pairs between human and mouse in HOVERGEN release 46 found 9144 families in which 13 233 orthologs have been identified.

DISCUSSION AND CONCLUSION

Tree rooting

It is important to note that reconciliation algorithms require that the trees are correctly rooted. Since phylogenetic inference methods produce unrooted trees, these trees will have to be rooted before reconciliation. One possibility consists in using the midpoint procedure: assuming a molecular clock, the roots correspond to the point that is at equal distance from all tree leaves. It is however clearly established that the molecular clock assumption is often incorrect, notably in multigenic families where paralogous genes can be subject to different selective pressures.

Another solution consists in defining an outgroup. For example, in a set of orthologous genes, the outgroup is constituted by the genes corresponding to the clade that diverged first in the species tree. Thus, a tree of orthologous genes can easily be rooted, provided one has some a priori knowledge about the most basal taxa in the species tree. However, in a phylogenetic tree containing paralogous genes, defining an outgroup requires first identification of duplication nodes, and hence cannot be done independently of the tree reconciliation.

As suggested by Zmasek and Eddy (2001), a parsimonious solution consists in placing the root in the gene tree so as to maximize the similarity between the gene tree and the species tree. Thus, the procedure we use to root our gene trees consists in using the reconciliation algorithm described above, to explore all possible positions of the root in the gene tree, and retain the position that requires the minimal number of gene duplications. In case of equality, we retain the candidate that is closest to the tree midpoint.

Identifying speciation and duplication events

In order to identify speciation and gene duplication events in a gene tree, it is necessary to compare gene and species trees. As already mentioned, several algorithms dealing with that problem have been previously described (Page and Charleston, 1997; Eulenstein *et al*., 1998; Ma *et al*., 2000; Zmasek and Eddy, 2001), but in many cases they are not suitable for real data because they require that both trees be completely resolved. Thus, any error in one of the trees will result in overestimation of duplication events. The RAP algorithm is able to cope with uncertainties, both in the gene and species trees. A node in the gene tree is considered as corresponding to a speciation event, as far as there is no strong evidence that the gene and species trees are incongruent. Moreover, RAP is able to take into account not only the topology of the trees, but also their bootstrap values or branch lengths. Finally, RAP is also an efficient method to identify the most parsimonious root in a gene tree. Another advantage provided by our algorithm is the fact that it is rapid enough to be used for the reconciliation of very large sets of phylogenetic trees. For example, the reconciliation of the 9926 phylogenetic trees of HOVERGEN release 46 containing at least three genes took $\sim$8.5 h on a 950 MHz SPARC processor.

A problem is that RAP does not weight gene losses because no cost is associated to them. The only parameters that influence the reconciliation are the tree topologies, and the branches lengths or bootstraps. In fact, in a database such as HOVERGEN, genome sequences are often incomplete and, consecutively, gene losses cannot be weighted relevantly. On the other hand, as HOGENOM contains only complete genomes, losses in reconciled trees from this database could be considered as real losses, and they can be weighted.

Another limitation of the tree reconciliation procedure proposed here is that it assumes that gene transmission has been entirely vertical. In animals, this assumption can be considered as correct because there are very few known cases of horizontal transfers in animals, and they are all related to transposable elements (Kordis and Gubensek, 1998). In prokaryotes, however, horizontal transfers may be relatively frequent (Ochman *et al*., 2000; Garcia-Vallvé *et al*., 2000; Koonin *et al*., 2001), although this issue is still heavily debated (Daubin *et al*., 2002, 2003). Nevertheless, we have used RAP to reconciliate HOBACGEN and HOGENOM gene trees, even if, in this case, the number of duplications inferred is probably overestimated. However, note that even in the absence of tree reconciliation, or if it is not trusted, it is still possible to automatically search for orthologs in these databases by using the tree pattern search facility.

In theory, taking branch lengths into account in RAP tends to underestimate the distance between genes. Consequently, in a monogenic family with some hidden paralogies, it may occur that RAP collapses some nodes and wrongly labels a duplication node as a speciation. However, a RAP parameter, the maximum length to collapse a node, can be corrected in order to take into account this underestimation of distances.

Search for tree patterns

Compared with a manual search, tree pattern matching presents advantages and inconveniences. The manual expertise allows one to consider existing anomalies on phylogenetic trees and brings a better flexibility in the search. Indeed, even if the formulation is very rich, an automatic request system can never satisfy perfectly the initial objective of the biologist. The algorithm is also sometimes dependent on reconstruction artefacts, in particular badly chosen phylogenetic roots for deep patterns. On the other hand, the tree pattern matching is a very fast operation. Searching for a pattern on an entire tree database is well compatible with an interactive application. HOVERGEN and HOBACGEN each contain about 10 000 trees, and a pattern search into one of these databases takes $\leq$30 s on our server.

Automatic search for orthologs

Presently, the most frequently used approach to automatically searching for orthologous genes in different species consists in searching for sequence similarities by pairwise alignments and then selecting the best reciprocal hits: if genes X from species A and Y from species B are orthologous, then one expects that in the genome B, Y be the closest homolog of X, and reciprocally, that in the genome

A, *X* be the closest homolog of *Y*. Thus, one can automatically search for orthologs between *A* and *B*, simply by comparing all their proteins between each other (e.g. with BLAST). This approach can be extended to more than two species by searching for a subset of best reciprocal hits among homologous genes, and was used for the Cluster of Orthologous Groups (COGs) database (Tatusov *et al*., 2001). An extension of this method has been developed to distinguish orthologs, in-paralogs and out-paralogs (paralogs that predate the species split) (Remm *et al*., 2001). An important limitation of these methods is that they can be used only for species for which all genes have been identified. Moreover, even when genomes have been entirely sequenced, this approach may give erroneous results because of variations in evolutionary rates within a gene family, or because some genes have been lost during evolution, or missed during the annotation process. This is more likely to happen in higher eukaryotes, where gene prediction is very difficult.

An important point that has to be highlighted is that the classification of genes into clusters of orthologs depends on the evolutionary distance between the species that are considered. Let us consider three taxa $T1$, $T2$ and $T3$. A set of homologous genes between $T1$ and $T2$ corresponds to a cluster of orthologs if and only if all these genes descend from a single gene in the last common ancestor of $T1$ and $T2$. Thus, the set of clusters of orthologs between taxon $T1$ and taxon $T2$ corresponds to the set of genes that were present in the genome of their last common ancestor (minus genes that have been lost in one lineage or the other). Hence, if the last common ancestor of $T1$ and $T2$ is different from the last common ancestor of $T1$ and $T3$, then the set of clusters of orthologs between $T1$ and $T3$ may differ from the set of clusters of orthologs between $T1$ and $T2$. The classification proposed in the COGs database is therefore not valid for all sets of taxa. In other words, the classification of genes into clusters of orthologs should be recomputed according to the taxa that are being considered. Figure 5 illustrates an example of this problem with the phylogenetic tree of a hypothetical gene family containing sequences from human, *Drosophila* and chicken. In this example the *X* and *Y* genes are paralogous, and result from a duplication predating the divergence between vertebrates and insects. These genes have undergone several duplications in *Drosophila* (*Ya*, *Yb*) and vertebrates (*Yv1*, *Yv2*, *Xv1*, *Xv2* and *Xv3* genes). Such a situation is very common in vertebrates, and might result from one or two genome duplications at the basis of this lineage. If one is interested in identifying all orthologs between mammals and birds, then one should classify *Yv1*, *Yv2*, *Xv1*, *Xv2* and *Xv3* into five distinct clusters (within each cluster, human and chicken genes are orthologous, but genes from different clusters are paralogous). But if one wants to identify all orthologs between *Drosophila* and vertebrates, then there should be only two clusters: *X*, *Xv1*, *Xv2* and *Xv3* should be in one cluster (since the *Drosophila X* gene is orthologous to *Xv1*, *Xv2* and *Xv3*); and *Ya*, *Yb*, *Yv1*, and *Yv2* should be in another cluster (since the *Drosophila Ya* and *Yb* genes are orthologous to both *Yv1* and *Yv2*). Note that the orthology relationship is not necessarily one-to-one: because of gene duplications having occurred after the divergence of species that are being considered, one gene in a given taxon may have several orthologs in another taxon (Sonnhammer and Koonin, 2002).

More recently, Zmasek and Eddy (2001, 2002) developed a more rigorous procedure that directly relies on the comparison of gene and species trees to automatically infer orthology relationships. The approach we propose is comparable and is more general. First, our reconciliation program is applied to whole gene families databases and not only on a limited number of genes and species for which one wants specifically to identify orthologs. Second, a dedicated graphical interface has been developed in order to facilitate the composition of queries. This is important because it makes it possible to build complex queries containing a lot of constraints on the branches and the nodes. Third, the tree pattern-matching algorithm itself is not limited to queries that allow the identification of orthologs as any kind of pattern can be entered.

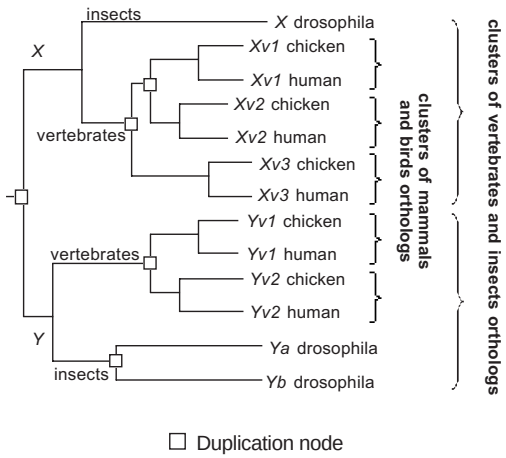

Fig. 5. An example of complex orthology relationships within a hypothetical gene family. *X* and *Y* are the duplicated copies of a gene in the common ancestor to vertebrates and insects. No duplication event occurred for *X* in the lineage leading to present day *Drosophila* species, but different duplications happened for *X* in vertebrates and for *Y* either in insects or in vertebrates.

However, it should be mentioned that the quality of the orthology inferences depends on the reliability of the phylogenetic tree; hence the rate of false positive or false negative depends on the evolutionary distances between the species of interest. Linked to that, gene families in which there is not enough phylogenetic information (such as homeobox containing genes) will give erroneous results and should be removed. On the other hand, as we only integrate in a given family the sequences that can be aligned on 80% of their length, the problem of poorly aligned sequences—leading to erroneous phylogenetic reconstructions—is avoided (Perrière *et al*., 2000). A possible improvement of RAP would be to provide an assessment of the reliability of the reconciliation. Storm and Sonnhammer (2002) proposed a procedure based on the analyses of bootstrapped trees: the results of reconciliation of each bootstrap tree are combined to give support levels of orthology inferences.

ACKNOWLEDGEMENTS

This work has been supported by EEC, CNRS and INRIA. J.F.D. was a recipient of a fellowship from INRIA. S.P. was a recipient of a fellowship from the EEC under the specific RTD program ‘Quality of Life and Management of Living Resources’, contract number QLRI-CT-2001–00015 for TEMBLOR.

REFERENCES

Aho,A.V. *et al*. (1989) Code generation using tree matching and dynamic programming. *ACM Trans. Program. Lang. Syst.*, **11**, 491–516.

Altschul,S.F. *et al*. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Daubin,V. *et al*. (2002) A phylogenomic approach to bacterial phylogeny: evidence of a core of genes sharing a common history. *Genome Res.*, **12**, 1080–1090.

Daubin,V. *et al*. (2003) Phylogenetics and the cohesion of bacterial genomes. *Science*, **301**, 829–832.

Duret,L. *et al*. (1999) HOVERGEN: database and software for comparative analysis of homologous vertebrate genes. In Letovsky,S. (ed.), *Bioinformatics Databases and Systems*. Kluwer Academic Publishers, Boston, pp. 13–29.

Eulenstein,O. *et al*. (1998) Duplication-based measures of difference between gene and species trees. *J. Comput. Biol.*, **5**, 135–148.

Garcia-Vallvé,S. *et al*. (2000) Horizontal gene transfer in bacterial and archaeal complete genomes. *Genome Res.*, **10**, 1719–1725.

Kilpeläinen,P. (1992) Tree matching problems with application to structured text databases. Departement of Computer Science. University of Helsinki, Helsinki.

Kilpeläinen,P. and Mannila,H. (1993) Retrieval from hierarchical texts by partial patterns. In Korfhage,R., Rasmussen,E.M. and Willett,P. (eds), *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York, pp. 214–222.

Koonin,E.V. *et al*. (2001) Horizontal gene transfer in prokaryotes: quantification and classification. *Annu. Rev. Microbiol.*, **55**, 709–742.

Kordis,D. and Gubensek,F. (1998) Unusual horizontal transfer of a long interspersed nuclear element between distant vertebrate classes. *Proc. Natl Acad. Sci. USA*, **95**, 10704–10709.

Lynch,M. *et al*. (2001) The probability of preservation of a newly arisen gene duplicate. *Genetics*, **159**, 1789–1804.

Ma,B. *et al*. (2000) From gene trees to species trees. *SIAM J. Comput.*, **30**, 729–752.

Ochman,H. *et al*. (2000) Lateral gene transfer and the nature of bacterial innovation. *Nature*, **405**, 299–304.

Page,R.D.M. and Charleston,M.A. (1997) From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Mol. Phyl. Evol.*, **2**, 231–240.

Perrière,G. *et al*. (2000) HOBACGEN: database system for comparative genomics in bacteria. *Genome Res.*, **10**, 379–385.

Remm,M. *et al*., (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, **314**, 1041–1052.

Sonnhammer,E.L. and Koonin,E.V. (2002) Orthology, paralogy and proposed classification for paralog subtypes. *Trends Genet.*, **18**, 619–620.

Storm,C.E. and Sonnhammer,E.L. (2002) Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics*, **18**, 92–99.

Tatusov,R.L. *et al*. (2001) The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Res.*, **29**, 22–28.

Wheeler,D.L. *et al*. (2004) Database resources of the National Center for Biotechnology Information: update. *Nucleic Acids Res.*, **32**, D35–40.

Zmasek,C.M. and Eddy,S.R. (2001) A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics*, **17**, 821–828.

Zmasek,C.M. and Eddy,S.R. (2002) RIO: analyzing proteomes by automated phylogenomics using resampled inference of orthologs. *BMC Bioinformatics*, **3**, 14.