The Rainbow Prim Algorithm for Selecting Putative Orthologous Protein Sequences

Anna Rosa Troisi[1], Giovanni Aloisio[1], Ivan Rossi[2], Piero Fariselli[2] and Rita Casadio*[,2]

[1]*University of Salento & SPACI Consortium, via per Monteroni 73100 Lecce, Italy*

[2]*Biocomputing Group - Department of Biology, University of Bologna, via Irnerio 42 40126 Bologna, Italy*

Abstract: We present a selection method designed for eliminating species redundancy in clusters of putative orthologous sequences, to be applied as a post-processing procedure to pre-clustered data obtained from other methods. The algorithm can always zero-out the cluster redundancy while preserving the number of species of the original cluster

INTRODUCTION

Protein sequences are defined as orthologous if they are separated by a speciation event: if a gene exists in a species, and that species diverges into two species, then the divergent copies of this gene in the resulting species are orthologous. Sequence orthology usually implies a common sequence function [1], and clustering of protein sequences to find orthologous chains is an essential step of comparative genomics.

All the clustering methods start with an all-against-all pairwise protein sequence similarity search. From this set of alignments, clusters are then built according to criteria such as merging triangles on the bidirectional-best-hit graph when they share a common side [2], graph-contraction hierarchical clustering [3-4], Markov Chain Clustering [5-6], normalized-cut graph partitioning [7], clustering that takes into accounts the known phylogenetic data [8-10]. Each cluster obtained should ideally contain a set of orthologous proteins that share a common function. This relation automatically yields a number of useful functional predictions for poorly characterized genomes.

According to the definition of orthology, at most one sequence for each species should be present for each cluster of putative orthologs (CPO), but unfortunately this is not always the case: most of the methods described in the literature, to some extent, generate clusters that contain more than a sequence per organism (they contain paralogs). Distinguishing between orthologs and paralogs has biological value, since only the orthologs are likely to show conserved function across different species. In this paper, we propose a procedure to help in separating orthologs and paralogs, starting from the results generated by the already-established methods. This procedure is designed to extract subclusters that do not have species redundancy and, at the same time, preserve the maximal similarity, in terms of the original adopted metric, among the selected sequences. Moreover, we designed our algorithm to be fast.

*Address correspondence to this author at the Biocomputing Group - Department of Biology, University of Bologna, via Irnerio 42 40126 Bologna, Italy; E-mail: casadio@biocomp.unibo.it

RESULTS AND DISCUSSION

Rainbow Prim Algorithm

The algorithm we proposed is based on a modification of the original Prim algorithm [11]. It looks for a minimum spanning tree (MST) embedded in the graph that describes the distances between the protein sequences belonging to a CPO, with an added condition that the set of the final vertices must contain only one type of "color" (or species), like in a rainbow (from this we derive its name of Rainbow Prim). In this way we guarantee that the final MST will represent a subset of the original cluster, where each organism is uniquely represented. The algorithm returns the smallest-weight spanning tree among the maximum-cardinality spanning trees found.

In order to increase the searching space of the Rainbow Prim, we apply the algorithm starting from all vertices whose color is not unique in the graph (all protein chains in the cluster whose originating specie is not uniquely represented) and the output we consider is the tree with the lowest weight among those of maximum cardinality generated by the algorithm; this set of vertices represents our new cluster.

The algorithm is completely deterministic and thus always gives a unique tree.

Rainbow Prim is a greedy algorithm that at each iteration optimizes the choice regardless to previous choices ("doing the best locally"). Although in general, greedy algorithms do not guarantee to find the optimal solution of a given problem, they are fast. Furthermore, the problem under examination, that is selecting K nodes in a graph in such a way that their spanning tree is minimal (minimal K-spanning tree) has NP complexity [12], and thus the only known method that guarantees the finding of the optimal solution is the complete enumeration, whose time requirement grows exponentially. Using a binary heap data structure and an adjacency list representation, Rainbow Prim's algorithm can be shown to run in time which is O((V-K)E log V) where E is the number of edges, K is be the number of colors/species in the cluster, and V is the number of vertices. Even more time-consuming than complete enumeration of the K-spanning trees would be to use methods based on computational phylogenetics, such as finding the lowest-weight K-leaves neighbore-joining

1875-0362/08

tree, that will require the evaluation of $O(K^{N/K})$ different phylogenetic trees, where K is the number of species and N is the cardinality of the considered ensemble.

Rainbow Prim Algorithm Pseudo Code

Rainbow Prim starts with a single vertex of graph G and at each iteration adds to the current tree a minimum-weight edge that does not complete a cycle. The following is a pseudo code version of Rainbow Prim algorithm.

If (x,y) is an edge in G = (V,E) then w(x,y) is its weight, otherwise $w(x,y) = \infty$. The starting vertex is s.

```
for each s in G do rainbow_prim(V, w, s);
    procedure rainbow_prim(V,w,s){
    V' := {s} // vertex set starts with s
    E' = {} // edge set initially empty
    C' ={c(s)} //colors set starts with s color
    for i := 1 to n-1 do // put n edges in spanning tree
    {
        find x in V', y in V-V'with MIN(w(x,y)) AND color(y) NOT in C'
        add y to V'
        add c(y) to C'
        add (x,y) to E'
    }
    return(E')
    }
```

Rainbow Prim Testing

Since Rainbow Prim by construction guarantees the complete removal of the redundancy from any cluster, it becomes interesting to evaluate the performance in terms of "missing species" with respect to the original ensemble. In other words, if the original CPO contains Np proteins divided into Ns species (with Np>Ns), after the run of Rainbow-Prim we obtain a tree that consists of NRp proteins. By construction, NRp is also the number of species kept into the set. However, Rainbow-Prim does not guarantee that NRp is equal to the original number of the species into the starting CPO (Ns). We can quantify the original redundancy as R(original)=Np/Ns where Np is the number of proteins and Ns is the number of species, and the fraction of missing species as R(rainbow prim)= NRp/Ns, where Nrp is the number of proteins left after the Rainbow Prim.

Data Set

We started using the DomClust COG03 data set [3,13] comprising 4813 different clusters from 66 bacterial genomes. We chose this data set as our starting point, because the author already used it as a benchmark to measure, among other features, the ability of some methods to produce redundancy-free clusters (see Table **2** of Ref. [3]). However, we excluded from our analysis all the clusters that have the following characteristics:

- perfectly non-redundant; in this group of 1724 CPOs, for each cluster the number of species matches the number of contained proteins. There CPOs truly represent the perfect definition of Cluster of Orthologous sequences, thus, no further processing is necessary.
- unconnected CPOs (399); in this case, each cluster cannot be represented as a connected graph (but as a forest). In this case, it is clearly impossible to define a spanning tree.
- clusters containing multidomain proteins that are represented using two different nodes in the cluster, one for each domain (79).

After this filtering procedure, we ended up with a working set that consists of 2611 redundant clusters. On this set, each cluster is represented as a colored weighted graph where each protein is represented by a vertex, and whose "color" represents the species to which the protein belongs. The weight of the edges represent the distance between two protein sequences and the value is taken as a function of the BlastP [14] E-value. In particular, the distance measure, described in ref [3], is taken as symmetrical and is computed as:

$\text{Min}(\text{dist}(A,B), \text{dist}(B,A))$;

where:

$\text{dist}(X,Y) = E \; lx ly 10^{-5}$

E: Blastp E-value calculated using a fixed search space size of 10^9

lx,ly: length of the aligned portions of the query and of the hit

As we can see from Fig. (**1**), where R(original) and R(Rainbow Prim) are plotted for each CPO in our working set, the Rainbow-Prim performance is quite good and in 2607 out of 2611 cases (99.8%) it reaches the complete coverage of the all available species (a value of one in the graph shown in Fig. **1**), while at the same time it completely eliminates the corresponding CPO redundancy.

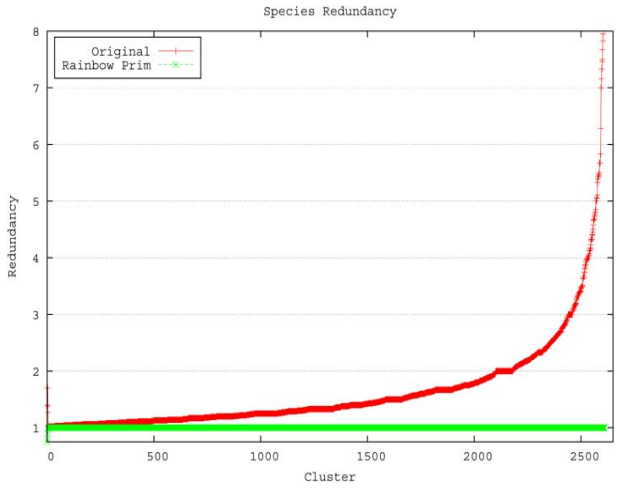

Fig. (1). Redundancy of DomClust clusters (Original) vs Redundancy of the corresponding Rainbow-Prim-processed clusters (Rainbow Prim).

It is worth noticing that, for the few cases where the complete coverage was not achieved, the problem is due to the

fact that the underlying graph has a structure of a "bipartite graph". In particular, even though both partitions contain redundant colors, it is not possible to build a tree that spanned the missing species separated into the two partitions at the same time (e.g. see Fig. **2**).

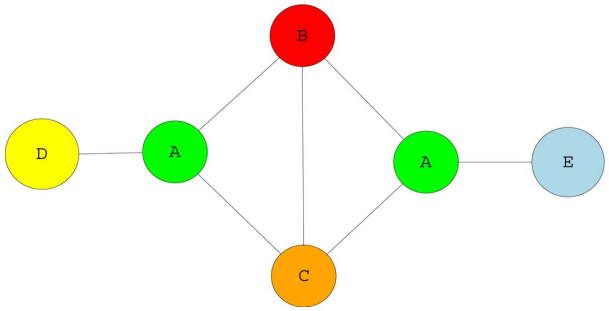

Fig. (2). An example of a CPO where Rainbow Prim cannot achieve complete species coverage. Either species ABCD or ABCE may be part of the same spanning tree.

Comparison with Another Simple Greedy Strategy

We also compared Rainbow Prim to another very simple greedy selection scheme that guarantees redundancy elimination, called Rainbow Simple. The following is a pseudo code version of Rainbow Simple algorithm.

```
Procedure rainbow_simple
for each node N in graph:
    score(N)= Sum of the weight of the outgoing arcs
partition the node set in classes by color
rainbow=empty
for each classes:
    rainbow.add(node N with minimum score )
return rainbow
end rainbow_simple
```

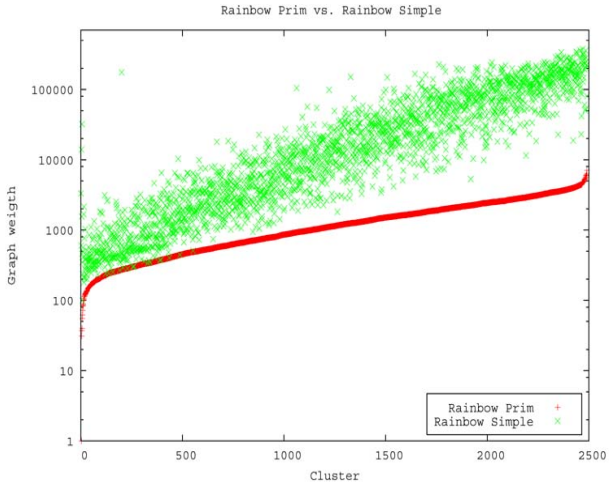

Fig. (3). Rainbow Prim vs Rainbow Simple: plot of the weights for the connected graphs generated by the Rainbow Prim (black dots) and the Rainbow_simple (purple dots) algorithm, starting from the same original DomClust cluster.

It turns out that the Rainbow_Simple procedure does not guarantee that the resulting subset of selected elements are still connected by bidirectional best hit relationship (the resulting graph is not a connected component of the original one). In our DomClust test set, it fails in 199 cases out of 2611 (4.6%), but, more importantly, when it succeeds, it always produces trees with higher (or at best equal) weights than Rainbow Prim, as shown in Fig. (**3**).

CONCLUSION

In the present version, Rainbow Prim does not offer a complete recipe to generate clusters of orthologous sequences starting from whole genomes. However, due to its speed, performance, and ease of implementation, we think that it can be a useful post-processing addition to most of the ortholog clustering methods already established.

ACKNOWLEDGEMENTS

GA and RC acknowledge the receipt of the grant FIRB 2003 LIBI-International Laboratory of Bioinformatics. RC also acknowledges the support to the Bologna node of the Biosapiens Network of Excellence project within the European Union's VI Framework Programme (contract number LSGH-CT-2003-503265).

All the large-scale testing computations have been performed using the Computational Grid at SPACI.

REFERENCES

[1] E.V. Koonin, "An apology for orthologs — Or brave new memes", *Genome Biol.*, vol. 2, comment1005.1-1005.2, April 2001.

[2] R.L. Tatusov, E.V. Koonin, and D.J. Lipman, "A genomic perspective on protein families", *Science*, vol. 278, pp. 631-637, October 1997.

[3] I. Uchiyama, "Hierarchical clustering algorithm for comprehensive orthologous-domain classification in multiple genomes", *Nucleic Acids Res.*, vol. 34, pp. 647-658, January 2006.

[4] R. Petryszak, E. Kretschmann, D. Wieser, and R. Apweiler, "The predictive power of the CluSTr database", *Bioinformatics*, vol. 21, pp. 3604-3609, September 2005.

[5] A.J. Enright, S. Van Dongen S, and C.A. Ouzounis, "An efficient algorithm for large-scale detection of protein families", *Nucleic Acids Res.*, vol. 30, pp. 1575-1584, April 2002.

[6] L. Li, C.J. Stoeckert Jr., and D.S. Roos, "OrthoMCL: identification of ortholog groups for eukaryotic genomes", *Genome Res.*, vol. 13, pp. 2178-2189, September 2003.

[7] F. Abascal, and A. Valencia, "Clustering of proximal sequence space for the identification of protein families", *Bioinformatics*, vol. 18, pp. 908-921, July 2002.

[8] A. Alexeyenko, I. Tamas, G. Liu, and E.L. Sonnhammer "Automatic clustering of orthologs and inparalogs shared by multiple proteomes", *Bioinformatics*, vol. 22, pp. e9-e15, July 2006.

[9] D.L. Fulton, Y.Y. Li, M.R. Laird, B.G. Horsman, F.M. Roche, and F. S. Brinkman, "Improving the specificity of high-throughput ortholog prediction", *BMC Bioinformatics*, vol. 7, pp. 270, May 2006.

[10] A. Vashist, C.A. Kulikowski, and I. Muchnik, "Ortholog clustering on a multipartite graph", *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 4, pp. 17-27, January 2007.

[11] R.C. Prim, "Shortest connection networks and some generalisations", *Bell Syst. Tech. J.*, vol. 36, pp. 1389-1401, 1957.

[12] C. H. Papadimitriou, M. Yannakis "The complexity of Restricted Spanning Tree Problems", *J. ACM*, vol. 29, pp. 285-309, April 1982.

[13] DomClust: Hierarchical Clustering Program for Orthologous Protein Domain Classification. [Online] Available: http://mbgd.nibb.ac.jp/domclust/cog03.tgz

[14] S. A. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman "Basic Local Alignment Search Tool", *Mol. Biol.,* vol. 215, pp. 403-410, October 1990.

Received: July 10, 2008 Revised: July 29, 2008 Accepted: August 08, 2008