BIOINFORMATICS APPLICATIONS NOTE Vol. 30 no. 5 2014, pages 734–736
doi:10.1093/bioinformatics/btt582

Sequence analysis Advance Access publication October 9, 2013

orthAgogue: an agile tool for the rapid prediction of orthology relations

Ole Kristian Ekseth*, Martin Kuiper and Vladimir Mironov

Semantic Systems Biology Group, Department of Biology, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway

Associate Editor: John Hancock

ABSTRACT

Motivation: The comparison of genes and gene products across species depends on high-quality tools to determine the relationships between gene or protein sequences from various species. Although some excellent applications are available and widely used, their performance leaves room for improvement.

Results: We developed orthAgogue: a multithreaded C application for high-speed estimation of homology relations in massive datasets, operated via a flexible and easy command-line interface.

Availability: The orthAgogue software is distributed under the GNU license. The source code and binaries compiled for Linux are available at https://code.google.com/p/orthagogue/.

Contact: orthagogue-issue-tracker@googlegroups.com

Received on February 26, 2013; revised on September 11, 2013; accepted on October 4, 2013

1 INTRODUCTION

The notion of sequence homology in biology refers to similarities at the nucleotide or amino acid sequence level that may reflect ancestry and evolutionary conservation. Two types of homology are most often distinguished: orthology (resulting from a speciation event) and paralogy (resulting from a gene duplication). Software tools for establishing homologies between genes or their products are becoming increasingly important to transfer knowledge from well-studied model organisms to other organisms (Altenhoff and Dessimoz, 2012).

Identification of genuine homology relationships is no trivial task, as it requires the knowledge of the last common ancestor (LCA) for every pair of sequences. In theory, LCAs could be predicted on the basis of phylogenetic analysis but this is not feasible at the proteome scale due to performance limitations. Therefore, a variety of heuristic approaches are used instead of exhaustive phylogenetic analyses to estimate orthology relations. Depending on the heuristics, different sets of orthologs are generated with a varying degree of overlap and it is not trivial to gauge the precision of predictions objectively. As a result a large number of widely used database resources [e.g. COG (Tatusov *et al*., 2000), KOG (Tatusov *et al*., 2003), eggNOG (Jensen *et al*., 2008), Roundup (DeLuca *et al*., 2006) and OMA (Altenhoff *et al*., 2011)] were generated.

Nevertheless, as the number of complete proteomes is steadily growing, even heuristic algorithms meet their performance limitations, an issue that is bound to become more and more acute in the future. Therefore, we embarked on a high-performance implementation of one of the popular heuristic approaches exemplified by the OrthoMCL software (Chen *et al*., 2006; Li *et al*., 2003).

The OrthoMCL procedure comprises three major steps:

(1) Performing an all-against-all BLAST (Basic Local Alignment Tool, Altschul *et al*., 1990).
(2) Identification of putative orthology and inparalogy relations with the Inparanoid algorithm (Remm *et al*., 2001).
(3) Generation of disjoint clusters of closely related proteins with the Markov Clustering Algorithm (MCL) (implemented independently of OrthoMCL) (Enright *et al*., 2002).

Steps 1 and 2 tend to be time-consuming with large datasets comprising complete proteomes (in the range of a couple of days each for 150 proteomes, the exact duration depending on the setup). While the time constraints of step 1 can be mitigated by deployment on a large computer cluster, there is no way to shorten step 2.

We therefore re-implemented step 2 (comprising sub-steps 8–11 of the OrthoMCL procedure as outlined in its user guide (http://orthomcl.org/common/downloads/software/v2.0/UserGuide.txt), and developed the C application orthAgogue (from the Greek 'agogos' = 'lead'). The development of this application implied boosting the data parsing efficiency, optimizing the memory use and data containers and providing a flexible command line user interface. The result is an open source software with a significantly shorter running time and an increased algorithmic flexibility.

2 IMPLEMENTATION

We made use of multi-threading and highly efficient data containers. This was achieved by implementing orthAgogue in C with the Threading Building Blocks (TBB) library (http://software.intel.com/en-us/intel-tbb) and Message Passing Interface (MPI) (Graham *et al*., 2006) used for parallelization, and the C Minimal Perfect Hashing (CMPH) library (http://cmph.sourceforge.net/) for building efficient data containers. To provide more flexibility to the user, we devised a command line interface with multiple configuration options. In particular, these options can be used to configure orthAgogue to use one of the three methods for computing similarirty scores.

*To whom correspondence should be addressed.

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

© The Author 2013. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

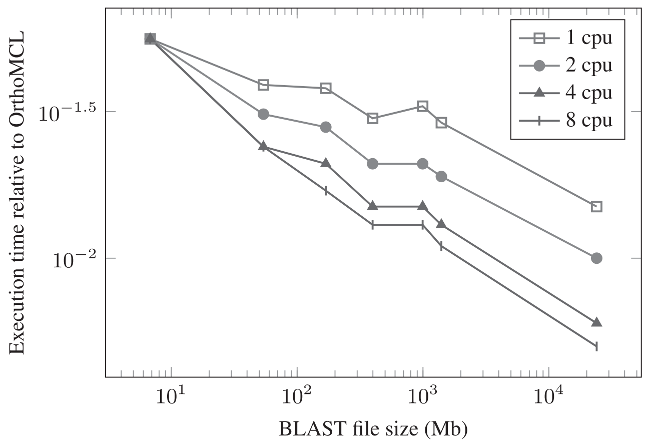

Fig. 1. Benchmarking of OrthAgogue and OrthoMCL using the imput BLAST files ranging from 6.8 MB to 24 GB. The performance of OrthAgogue is expressed as a fraction of the processing time of OrthoMCL. The number of Central Processing Units (CPU) used is indicated in the top right corner

The output of orthAgogue comprises three sets of protein pairs (following the definitions of the Inparanoid algorithm):

(1) Orthologs: inter-species pairs with reciprocal best BLAST scores.
(2) Inparalogs: intra-species pairs with a better BLAST score than either protein of that pair has with proteins outside the taxon.
(3) Co-orthologs: for any given pair of orthologs, all possible ortholog–inparolog pairs across taxonomic borders (with the exclusion of the ortholog pairs).

The union of (1) and (3) provides the most plausible estimate of the set of true orthologs. The output consists essentially of three symmetric adjacency matrices serialized in two formats conducive to MCL: lists of homologous protein pairs (the format used in OrthoMCL, *.abc files) and native MCL matrices (the preferred input for MCL, *.mci files). The file proteins.map provides a mapping between the protein indices used in *.mci files and the strings (taxonID–proteinID) used in *.abc files. Optionally, the output can be piped directly into MCL (all.mci file).

3 RESULTS AND DISCUSSION

To illustrate the considerable reduction in processing time, we benchmarked orthAgogue against OrthoMCL 2.0.3 with a set of seven BLAST outputs ranging in size from 6.8 Mb to 24 Gb. The result (Fig. 1) indicates that even with only a modest number of CPUs, orthAgogue performs up to 200 times faster than OrthoMCL. Importantly, the gain in performance steadily increases with the size of the dataset. For the complete set of Reference Proteomes (147 in total, release April 2013 http://www.ebi.ac.uk/reference_proteomes/) the processing times of orthAgogue and OrthoMCL are 10 min and 33.5 h, respectively. Other differences between orthAgogue and OrthoMCL are summarized in Table 1.

While orthAgogue follows the same Inparanoid algorithm as OrthoMCL, a couple of changes were made intended to provide some flexibility. It is obvious from the code of OrthoMCL scripts that only the best high-scoring pairs in BLAST output (HSP) for a given pair of sequences is used. We reasoned that the use of additional HSPs would help differentiate between otherwise identical sets of sequences. Therefore, orthAgogue by default makes use of all HSPs. However, with the option '-b' orthAgogue will use only the best HSP for a given pair of proteins. We also attempted to tackle a problem common to all applications using BLAST, namely the issue of '0.0' e-values in BLAST output. These result from the overflow of double precision variables. However, the alignment scores that are used by BLAST to compute e-values retain the differences and can be used to discriminate the levels of similarity. Optionally, orthAgogue can make use of these scores (to be found in the last column of the '−m 8' output of BLAST) instead of the e-values to compute the similarities. Some typical examples of using orthAgogue can be found at the bottom of the orthAgogue help message (while running 'orthAgogue' without arguments).

Table 1. Comparison of orthAgogue and OrthoMCL

Aspect	OrthoMCL	OrthAgogue	Reason
Treatment of HSPs	Best only	All or best only	Flexibility
Number of steps	4	1	Ease of use
Need for RDBMS	Yes	No	Ease of use
FASTA header format	Rigid	Flexible	Ease of use
Use alignment scores instead of e-values	No	Yes	Problem of '0.0' e-values
Symmetric matrix output	No	Yes	Compatibility with MCL
Native MCL matrix output	No	Yes	Compatibility with MCL

Note: HSP, high-scoring pairs in BLAST output; RDBMS, relational database management system.

We conclude that orthAgogue is an extremely efficient and flexible tool that can benefit every life scientist interested in the prediction and use of orthology information.

ACKNOWLEDGEMENTS

The authors thank the High Performance Computing Group at NTNU for their support.

Conflict of Interest: none declared.

REFERENCES

Altenhoff,A.M. and Dessimoz,C. (2012) Inferring orthology and paralogy. In: *Evolutionary Genomics*. Humana Press Inc., NY, pp. 259–279.

Altenhoff,A.M. *et al.* (2011) Oma 2011: orthology inference among 1000 complete genomes. *Nucleic Acids Res.*, **39**, D289–D294.

Altschul,S. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Chen,F. *et al.* (2006) OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Res.*, **34** (Database issue), D363–D368.

DeLuca,T.F. *et al.* (2006) Roundup: a multi-genome repository of orthologs and evolutionary distances. *Bioinformatics*, **22**, 2044–2046.

Enright,A.J. *et al.* (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, **40**, 1575–1584.

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

Graham,R.L. *et al.* (2006) Open mpi: a flexible high performance mpi. In: *Parallel Processing and Applied Mathematics*. Springer-Verlag, Berlin, Heidelberg, pp. 228–239.

Jensen,L.J. *et al.* (2008) eggNOG: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Res.*, **36**, D250–D254.

Li,L. *et al.* (2003) OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Res.*, **13**, 2178–2189.

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

Tatusov,R.L. *et al.* (2000) The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.*, **28**, 33–36.

Tatusov,R.L. *et al.* (2003) The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, **4**, 41.

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016