|  |
| --- |
| Genome Analysis  PorthoMCL: Parallel orthology prediction using MCL for the realm of massive genome availability  Ehsan S. Tabari1, and Zhengchang Su1\*  1Department of Bioinformatics and Genomics, The University of North Carolina at Charlotte, 9201 University City Blvd, Charlotte, NC 28223.  Received on XXXXX; revised on XXXXX; accepted on XXXXX  Associate Editor: XXXXXXX |

[[1]](#footnote-2)\*abstract

**Motivation:** Finding orthologous genes among multiple sequenced genomes is a primary step in comparative genomic studies. With the availability of exponentially increasing number of sequenced genomes, comparative genomics becomes more powerful than ever for genomic analysis. However, the very large number of genomes needing to be analyzed makes conventional orthology prediction methods incapable for the tasks. Thus a ultrafast tool is urgently needed.

**Results:** Here, we present PorthoMCL, an improved version of OrthoMCL with parallelization, for finding orthologous genes among a very large number of genomes.

**Availability:** PorthoMCL (source code, executables, sample datasets and documentation) is available under the MIT license in the github repository: [github.com/etabari/PorthoMCL](https://github.com/etabari/OrthoMCLP). The results of orthologs identified for 2,758 prokaryotic genomes are available for downloading at: UPLOAD IT SOMEWHERE (10GB compressed [51gb uncompressed]).

**Contact:** zcsu@uncc.edu.

# introduction

The rapid advance in sequencing technologies has made sequencing a prokaryotic genome at an unprecedented fast speed and low cost. As a result, thousands of prokaryotic genomes have been fully sequenced, and this number can soon reach tens of thousands. The availability of large number of completed genomes renders comparative genomics ever a powerful approach for gene annotations and addressing many important theoretical and application problems. However, the rate at which genomes are sequenced outpaces that at which CPU speed increases. This poses a great challenge in comparative analyses of the very large number of genomes, soliciting new faster algorithms or adapting existing tools in parallel environments.

Orthologs are genes in different species that are derived from a single gene in their last common ancestor by speciation events. Orthology indicates the conservation in both sequence and function between genes in different genomes. In contrast, paralogs are genes that are resulted from gene duplication within a species, thus may have different functions though their sequences can be conserved. Depending on the duplication happened before or after speciation, they are called outparalogs or inparalogs, respectively (Sonnhammer *et al.*, 2002). Therefore, identification of orthologous genes among a group of genomes is crucial to almost any comparative genomic analysis (Alexeyenko *et al.*, 2006). However, a major challenge in orthologs predictions is to differentiate true orthologs of a gene from the orthologs of its paralogs.

OrthoMCL is one of the most widely used algorithms for predicting orthologous genes across multiple genomes. Similar to many other orthology prediction algorithms, it is based on reciprocal best hits in all-against-all BLAST searches of complete proteomes of the genomes. OrthoMCL represents the similarity among the sequences using a weighted graph, where nodes are the genes, and two genes and from genomes and respectively, are connected by an edge if x and y are a pair of reciprocal best hits with a similarity greater than a cutoff. Specifically, the weight of the edge is a normalized score () based on the E-values of the reciprocal hits, and is defined as,

(1)

(2)

To distinguish paralogs and orthologs, OrthoMCL identifies recent paralogs to be included in orthologous groups as within-species BLAST hits that are reciprocally better than between-species hits (Li *et al.*, 2003). Orthologs and paralogs are then identified by finding the heavily connected subgraphs using the Markov Clustering algorithm (Van Dongen, 2000). However, OrthoMCL relies on a relational database system for storing the data, finding reciprocal best hits and scoring them, making it especially inefficient when the number of genomes becomes large. To overcome this problem and to further speed up the method, we developed a parallel orthology prediction tool using MCL, PorthoMCL. In addition to the parallelization, a more efficient data structure makes PorthoMCL ultrafast and highly scalable. Furthermore, ,thusu and cloud computing platforms

# POrthoMCL

## Workflow

The workflow of PorthoMCL is similar to that of OrthoMCL (Figure 1). However, instead of depending on an external database server, PorthoMCL uses a distributed sparse matrix for more efficient data storage and retrieval. In addition, we parallelized all the computationally intensive steps of OrthoMCL. First, PorthoMCL performes all-against-all BLAST searches in parallel by dividing

query genomes in multiple groups, and then collates the results in the matrix. and parallelizes OrthoMCL’s main step, Find Pairs, in which reciprocal best hits are identified, scored and categorized to be orthologs or paralogs. PorthoMCL’s Find Pairs sub-steps are designed to be executed in parallel on a variety of high performance computing (HPC) environments. They are scalable and can exploit the capacity available to the HPC. However, these sub-steps are not independent and they need to be run in the designed order of execution. Each sub-step builds on the top of the previous steps. The detail of these sub-steps are:

PairsBestHit: keeps only the highest scored hits of a gene from one genome to others.

1. PairsOrthologs: looks for reciprocal hits between different genomes with a user-defined threshold of BLAST score, and lists them as orthologs. A normalized score is calculated for each orthology relationship. This step requires data generated in step (1).
2. PairsInParalogs: It looks for reciprocal hits in a genome and checks if the hits pass a user-defined threshold and are better than all the Orthologs, and lists them as inParalogs. A normalized score is calculated for each relationship. This step requires data generated in steps (1) and (2).
3. PairsCoOrthologs: It finds all pairs of genes across two genomes that are connected through ortholog and inParolog relationships. This step requires data generated in steps (2) and (3).

Find Pairs

orthomcl  
AdjustFasta

orthomcl  
FilterFasta

All-v-All

BLAST

orthomcl

BlastParser

porthomcl  
PairsBestHit

porthomcl  
PairsOrthologs

porthomcl  
PairsInParalogs

porthomcl  
PairsCoOrthologs

porthomcl

DumpPairFiles

MCL

orthomcl

MclToGroups

**Fig. 1.** Workflow of PorthoMCL. Blue boxes are original OrthoMCL steps, whereas PorthoMCL steps are in green. Dark blue boxes are the external applications that OrthoMCL requires. (When you make a color graph, always starts with Red, Green, Blue and Black, then others)

All-v-All

BLAST

The output of steps (2), (3) and (4) are used to create a sequence similarity graph that is then cut by the MCL program to predict orthologous, paralogous and co-orthologous gene groups. Execution of MCL for each output set is independent to one another. High performance computing support

PorthoMCL is designed to predict orthologs in an ever increasingly larger number of sequenced genomes in a high performance computing environments such as computing clusters or cloud computing platforms. We have included a TORQUE script with the package to facilitate its use in such environments. However, PorthoMCL also runs on a desktop or a server without the need for a database server which is advantageous over OrthoMCL.

# RESULTS

To illustrate the power of PorthoMCL, We have applied it to all the 2,758 sequenced bacterial genomes in GenBank (downloaded: April 2015) using their annotated protein sequences. These genomes contain a total of 8,661,583 protein sequences with a median length of 270 amino acids. They serve both as both the query and the database for all-against-all BLAST searches. After splitting the query into smaller files each containing about10,000 sequences, we used PorthoMCL’s parallelizing script to run BLAST searches (e-value cutoff: 1e-5; database size: 1e8). The combined output of the BLAST contained 2,957,375,578 hits. The total runtime of the BLAST searches were 11 days on a cluster with 60 computing nodes (each nodes has 12 cores and 36GBs of RAM), which would need 549 days if run on a single node. In the next step, PorthoMCL searched for reciprocal best hits and identified 850,273,323 ortholog gene pairs that formed 208,530 ortholog groups. While OrthoMCL could not finish this step after 35 days of running on a database server with 40 cores and 1TBs of RAM, PorthoMCL finished in 8 days.

The ortholog pairs (file size: 10GBs) and ortholog groups (file size: 51MBs) are available for download at UPLOAD IT SOME WHERE. Along with PorthoMCL, we have supplied a sample dataset for convenience. PorthoMCL does not require OrthoMCL to work. Options and arguments required at every step are discussed in detail in the documentation that accompanies PorthoMCL.

acknowledgements

We thank Jonathan Halter for his support and valuable contributions to this project.

*Funding*: This work was funded by the National Science Foundation (EF0849615 and CCF1048261) and NIH (R01GM106013).

*Conflict of Interest*: none declared.

References

Alexeyenko, A., Lindberg, J., Pérez-Bercoff, A., & Sonnhammer, E. L. L. (2006). Overview and comparison of ortholog databases. *Drug Discovery Today. Technologies*, 3(2), 137–43. doi:10.1016/j.ddtec.2006.06.002

Li, L., Stoeckert, C. J., & Roos, D. S. (2003). OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Research*, *13*(9), 2178–89. doi:10.1101/gr.1224503

Sonnhammer, E. L. ., & Koonin, E. V. (2002). Orthology, paralogy and proposed classification for paralog subtypes. *Trends in Genetics*, 18(12), 619–620. doi:10.1016/S0168-9525(02)02793-2

Van Dongen, S. (2000). Graph Clustering by Flow Simulation.

PhD thesis, University of Utrecht, Netherlands.

1. \*To whom correspondence should be addressed. [↑](#footnote-ref-2)