Broccoli

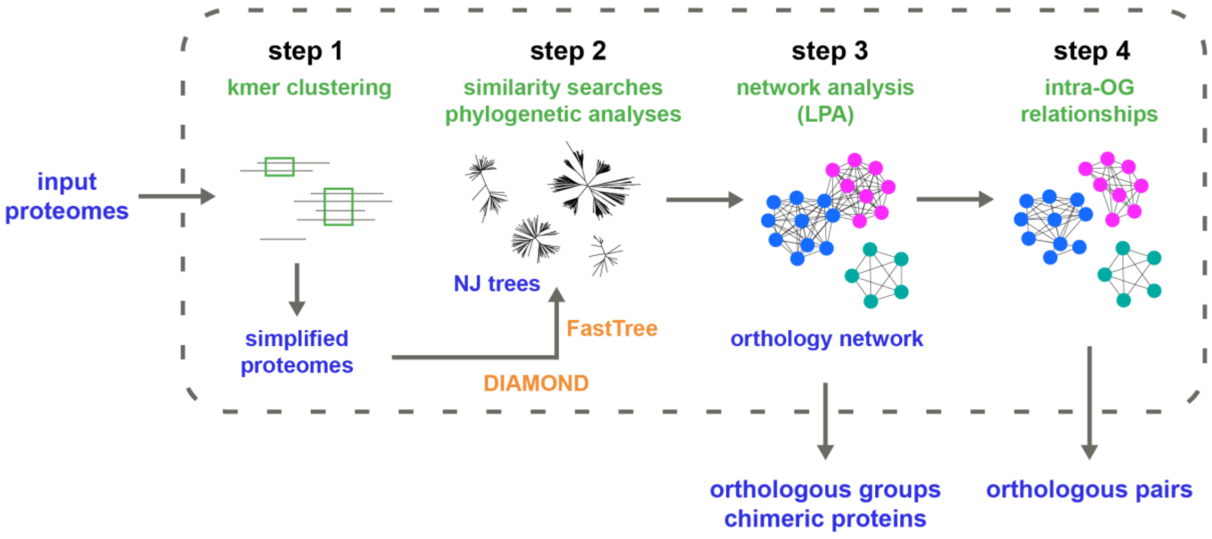
Manual - version 1.2

contact:
romain.derelle@gmail.com

Overview

Broccoli, is designed to infer with high precision orthologous groups and pairs of proteins using a mixed phylogeny-network approach. Briefly, Broccoli performs ultra-fast phylogenetic analyses on most proteins and builds a network of orthologous relationships. Orthologous groups are then identified from the network using a parameter-free machine learning algorithm (label propagation). Broccoli also detects chimeric proteins resulting from gene-fusion events and assigns these proteins to the corresponding orthologous groups.

Here is an overview of the 4 steps of the pipeline:



See manuscript: Broccoli: combining phylogenetic and network analyses for orthology assignment

Installation

You will need first to download and install DIAMOND 0.9.25+ and FastTree 2.1+ (the single-thread version). Please note that Diamond and FastTree can be installed locally (see Options step 2).

Then you will need to install Python 3.6+ and the ete3 library. This can be done for instance by creating a conda environment as follows:

1. install Conda in your home directory
2. create a conda environment named 'env-broccoli'

```
conda create -n env-broccoli python=3.6 ete3
```

3. activate the environment to run Broccoli

```
conda activate env-broccoli
```

Input data

The input of Broccoli consists of a set of proteome files grouped in a directory. Their format should be as follows:

- one proteome file per species
- sequences in FASTA format
- **the protein names extracted by Broccoli correspond to the string of characters located before the 1st space**. These protein names should be unique (if not, Broccoli prints a warning message).

Output data

Broccoli stores the temporary and output files in 4 directories named ‘dir_step1’ to ‘dir_step4’ corresponding to each of the 4 steps.

The main output files are:
- orthologous_groups.txt (in dir_step3)
- chimeric_proteins.txt (in dir_step3)
- orthologous_pairs.txt (in dir_step4)

Broccoli also generates at each step a log file in its corresponding directory, and a few statistic and table files are available in ‘dir_step3’.

Running Broccoli

The only mandatory argument is the name of the directory containing the proteome files. For instance, with the example dataset:

```
python broccoli.py -dir example_dataset
```

Broccoli will store the temporary and output files in 4 directories named dir_step1 to dir_step4 (one for each step) located in the current directory.

For improved precision, you can perform maximum-likelihood phylogenetic analyses (here only the first 3 steps and on 8 threads):

```
python broccoli.py -dir data -phylogeny ml -steps 1,2,3 -threads 8
```

To reduce the computational time on large datasets (i.e. 200 bacterial proteomes), you can reduce the kmer size (step1) and the number of hits per species (steps2):

```
python broccoli.py -dir data -kmer_size 60 -nb_hits 4 -threads 8
```

There are two things to keep in mind before running Broccoli:

1. to perform a given step, Broccoli requires all directories generated at the precedent steps

2. each step first deletes its corresponding directory if it already exists (to avoid using temporary files from a precedent run)

Options

general options

-steps	steps to be performed, comma separated (default = '1,2,3,4')
-threads	number of threads [default = 1]
-h, -help	show this help message and exit

The option -steps determines which step(s) of the pipeline will be executed. By default, all 4 steps are executed. For instance, if you only need orthologous groups as output and not orthologous pairs, you can limit the analysis to the 3 first steps: '-steps 1,2,3'.

The -threads option is used to indicate how many CPUs will be used by the pipeline. Note that there is virtually no additional gain to run Broccoli on more CPUs than the number of proteomes to be analysed.

step 1

-dir	name of the directory containing the proteome files [required]
-ext	extension of proteome files (default = '.fasta')
-kmer_size	length of kmers [default = 100]
-kmer_min_aa	min. nb of different aa a kmer should have [default = 15]

The input data is specified by the 2 parameters -dir (full or relative path of the directory) and -extension. Files with other extensions present in the directory will be ignored.

The option -kmer_size corresponds to the length of kmers, while -kmer_min_aa corresponds to the minimum number of different amino-acids a kmer should have (in order to avoid highly repetitive sequences).

step 2

-path_diamond	path of DIAMOND with filename [default = 'diamond']
-path_fasttree	path of FastTree with filename [default = 'fasttree']
-e_value	e-value for similarity search [default = 0.001]
-nb_hits	max nb of hits per species [default = 5]
-max_gap	max fraction of gap per position [default = 0.7]
-phylogenies	phylogenetic method: 'nj' (neighbor joining), 'me' (minimum evolution) or 'ml' (maximum likelihood) [default = 'nj']

By default, the 2 programs should be named 'diamond' and 'fasttree' and be present in your path. If installed locally, the relative/full path of these 2 programs should be given to Broccoli using the options -path_diamond and path_fasttree.

The -e_value and -nb_hits options corresponds to the options --evalue and --max-target-seqs of DIAMOND respectively.

The option -max_gap governs the trimming of alignments generated by Broccoli by allowing a certain fraction of gap per position.

step 3

-sp_overlap	max ratio of overlapping species in phylogenetic trees [default = 0.5]
-min_weight	min weight for an edge in the orthology network [default = 0.05]
-min_nb_hits	min number of hits belonging to the OG [default = 2]
-chimeric_shared	min fraction of connected nodes in each OG [default = 0.5]
-chimeric_nb_sp	min nb of species in OGs involved in gene-fusions [default = 2]

The option -sp_overlap controls the way orthologous groups are identified in phylogenetic trees. The ratio represents, at each node, the max. fraction of overlapping species within the ‘browsed leaves’ and within the ‘sister leaves’ (i.e. a relaxed version of the algorithm described in the MBE paper). Low values (e.g. 0.3) will result in small orthologous groups similar to those produced by Broccoli v1.0, while high values (e.g. 0.7) will result in larger orthologous groups similar to the orthogroups produced by OrthoFinder.

The option -min_weight allows to clean the orthology network from weak edges by removing all edges with low weights (by default edges with weight below 0.1). The main advantage is to speed step3 up.

The 3 remaining options control the behaviour of the 2 corrections applied after the identification of orthologous groups:

- spurious hits: -min_nb_hits (number *n* of DIAMOND hits a protein should have with other proteins of the same orthologous group; applied to orthologous groups of size >= *n* + 1)
- chimeric proteins: -chimeric_shared and -chimeric_nb_sp

step 4

-ratio_ortho	limit ratio ortho/total [default = 0.5]
-not_same_sp	ignore ortho relationships between proteins of the same species

The option -ratio_ortho controls the precision/recall of orthologous pairs: increasing the ratio improve precision while decreasing the ratio improve the recall of orthologous pairs.

Finally, -not_same_sp discards orthologous pairs of proteins from the same species (to be used for the Quest for Orthologs benchmark datasets).

FAQ

How to make Broccoli more precise (orthologous groups)?

Based on all the tests I have done, running Broccoli with minimum-evolution or maximum-likelihood phylogenies (step2) improves the delineation of closely related orthologous groups. However, these parameters increase by a factor 30x and 100x respectively the runtime of phylogenetic analyses compared to the default neighbour-joining.

Another option consists on lowering the parameter -sp_overlap (step3; e.g. a value of 0.4), which would improve precision but lower the recall (small orthologous groups missing some divergent in-paralogs). This option should be suitable for the identification of phylogenomic markers.

How to make Broccoli faster?

Broccoli is already fairly fast. But if you really need to reduce its runtime, there are 2 parameters you can play with:

- step1: reducing the kmer size allows to simplify further the proteomes and therefore reduce the overall computational time, but at the obvious risk of clustering paralogs together.

nb: reducing the kmer size has nearly no effect on the simplification of prokaryotic proteomes nb2: in some species, many unrelated proteins share identical fragments (e.g. viral domains in *Trichomonas vaginalis*)

- step2: reducing the number of hits per species reduces the number of sequences in alignments and trees, but potentially at the cost of a loss of precision (not yet fully investigated).

How much memory do I need to run Broccoli?

By far, the most memory demanding step of Broccoli is the beginning of step3, when all possible orthologous pairs are combined (up to hundreds of millions). Here are some examples of memory consumptions when running Broccoli with 8 threads under default parameters:

dataset	memory consumption (in GB)
19 eukaryotes	2.4
64 fungi	27
149 bacteria	48.9

If facing memory issues, you could (i) reduce the number of proteomes to analyse, or (ii) increase the simplification of proteomes at step1 by reducing the kmer size.

How to cite Broccoli?

If you use Broccoli in your research, please cite this paper (MBE, *in press*):

Broccoli: combining phylogenetic and network analyses for orthology assignment
Romain Derelle, Hervé Philippe, John K. Colbourne
bioRxiv 2019.12.13.875831; doi: https://doi.org/10.1101/2019.12.13.875831

If possible, please also cite the DIAMOND and FastTree articles.