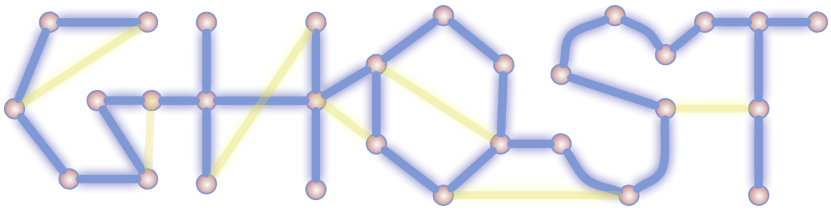

GHOST: User Guide v 0.5.0

Author: Rob Patro

Contents

Introduction

GHOST is a tool for the alignment of protein interaction networks. The technical details of the method as well as some results are described in the manuscript "Global network alignment using multiscale spectral signatures" (Patro and Kingsford 2012). This user guide provides a brief (and growing) overview of how to use the software tools that accompany the paper.

The process of aligning networks using GHOST can be viewed as a two-phase procedure. For every network one wishes to align, she must first extract the multiscale spectral signatures that are used to measure the topological similarity of vertices between different networks. This is done using the `ExtractSubgraphSignatures` command, the usage of which is detailed below in the Signature Extraction section. Note that the signatures themselves are meant to describe the local topology around a target vertex, and while specific to each network, they are independent of the specific pairwise alignment that is being performed. Thus, signatures need only be extracted once per network and can then be used in any number of pairwise alignments — there is no need to re-compute spectral signatures for different alignments.

The second step consists of using the extracted signatures for a pair of networks to perform the actual alginment. The description of how to run the network alignment step is detailed below in the Network Alignment section.

File Formats

Graph File Format

GHOST processes graphs in the `GEXF` file format. This is a general `xml`-based format for describing graphs. A full description of the `GEXF` format, along with libraries for reading `GEXF` formatted graphs in a number of languages can be found at http://gexf.net/format/.

Additionally, GHOST requires that graph vertices have a particular `string` attribute called 'gname' (short for gene name). Every vertex in a graph should have a unique `gname` attribute, and the `gname` string is how all vertices will be referred to in the output of the program (i.e. when computing signatures and performing alignments).
Specifically, the node attributes section of the file should look like the following:

```
<attributes class="node" mode="static">
  <attribute id="0" title="gname" type="string" />
  . . . # other node attributes
</attributes>
```

Then, the actual entry for a vertex in the graph will look something like the following:

```
<node id="35236" label="35236">
  <attvalues>
    <attvalue for="0" value="YMR061W" />
    . . . # other node attributes
  </attvalues>
</node>
```

Notice how the node has an `id` and `label` property that need not be the same as the `gname` property (here, the `gname` is `YMR061W`).

Sequence Distance File Format

GHOST can be given a secondary set of distances that it will additionally use when trying to determine which proteins should be mapped to eachother. The tradeoff between these distances and the spectral distances is either determined automatically or specified manually via the `-a`/`--alpha` parameter. These distances are most commonly derived from sequence comparison (e.g. BLAST e-values make a reasonable distance). If such distances are provided (via the `-x`/`--blast` option) each line in the file must be of the format:

```
a_x    b_y    d
```

for networks `A` = (V_A, E_A) and `B` = (V_B, E_B), where `A` is passed in as the first network (`-u`) and `B` is passed in as the second network (`-v`). `a_x` is the name of a node in network `A`, `b_y` is the name of a node in network `B`, and `d` is a numerical **distance** between the proteins (i.e. a smaller value denotes a smaller distance or greater similarity between the two proteins while a higher value denotes a greater distances or higher dissimilarity). The columns of this file should be tab-separated.

Alignment File Format

The alignments output by GHOST have a very simple format. For an alignment between networks `A` = (V_A, E_A) and `B` = (V_B, E_B), the GHOST alignment file is of the format:

```
a_1     b_1
a_2     b_2
.
.
```

```
.
a_m     b_m
```

where the first and second columns are separated by a tab character (`\t`). The file will contain `m` lines where `m` = min(|V_A|, |V_B|).

Spectral Distance Format

If the user runs GHOST with the `-d` command line option to dump the spectral distances between vertices in two different networks, `A` = (V_A, E_A) and `B` = (V_B, E_B) to a file, the distances are written to a file where each line has the following format:

```
a     b     d
```

Where `a` is the name of a vertex from network `A` , `b` is the name of a vertex from network `B` and `d` is the computed distance between the spectra of these vertices. The columns in this file are separated by a tab character (`\t`).

Signature Extraction

Note: The Scala-based signature extractor is deprecated. It is no longer supported and should not be used. Please use the C++-based signature extractor instead.

Before any network can be aligned, the spectral signatures for the vertices of this network must have been computed. Depending on the parameters chosen, this can be a time consuming and computationally expensive step. However, the spectral signatures are *alignment independent* and thus, this process need only be perfomed once per network.

Assume that the network for which we wish to extract the signatures is in the file `Net.gexf`. Signatures can be extracted using the following command:

```
> ./ExtractSignatures.sh -k 3 -p 10 -i Net.gexf -o Net.sig.gz
```

The options provided on the command line are as follows:

- **-k** The number of hops away from the central vertex for which each signature should be computed.
- **-p** The number of threads that should be used concurrently to extract signatures. Signatures for different vertices can be computed in parallel. Thus, the more threads are used, the faster the extraction can finish.

However, using more threads can also increase the memory usage, as each thread will need to build and decompose a matrix that represents the `k`-hop neighborhood of the vertex for which it is computing the signature.

- **-i** The network for which the signatures are to be computed. This network should adhere to the format described in graph format.
- **-o** The file where the signatures are to be written. The signatures are written in a compressed (gzipped) binary format.

Network Alignment

Networks are aligned using the executable `AlignNetworks.sh` script. This is a thin wrapper around an executable jar that does the real work. If you run `./AlignNetworks.sh --help`, you'll see a host of different options that can be passed into the executable. To shorten command line invocations and to help record the sets of parameters used for different alignments, GHOST is capable of accepting many alignment parameters from a configuration file in addition to from the command line directly. The parameters for the network alignment phase of GHOST are described below.

A particular run of the aligner might look something like the following:

```
> ./AlignNetworks.sh -c net1_vs_net2.cfg -o net1_vs_net2.aln -p 10
```

This aligns the networks specified in the configuration file `net1_vs_net2.cfg`, using the alignment parameters specified within. It will take advantage of up to 10 concurrent threads of execution, and the resulting alignment will be written to `net1_vs_net2.aln`.

General Parameters

The general parameters (i.e. those that refer specifically to the technical behavior of the program rather than the parameters of the alignment) are as follows:

- **-p | –numProcessors** This option accepts an integer argument; the maximum number of threads that should be run in parallel
- **-d | –dumpDistances** This option accepts a string argument. Instead of performing the actual network alignment, GHOST will simply compute all spectral between the vertices of `network1` and the vertices of `network2`, write them to a file with the provided name and then exit. The distances are written out in the format described in the Spectral Distance Format section.

Alignment Parameters

Alignment parameters affect the specific behavior of GHOST's network alignment algorithm. There are a numerous different parameters and they are described below. In addition to being able to provide these parameters on the command line, many of them can additionally be provided (perhaps via a slightly modified name) in a configuration file. A typical configuration file will look something like the following:

```
[main]
network1: Data/CJejuni/cjejuni.gexf
network2: Data/EColi/ecoli.gexf
sigs1: Data/CJejuni/cjejuni.sig.gz
sigs2: Data/EColi/ecoli.sig.gz
sequencescores: Data/CJejuni_vs_EColi.evalues
nneighbors: all
searchiter: 10
beta: 1.0
ratio: 8.0
```

The configuration file must start with the `[main]` section heading as shown above. The main section is followed by a list of key-value pairs, one per line, that designate different settings for alignment options.

Below we describe the avaialable parameters, how they are passed in as command line arguments (cmd) or via a configuration file (cfg), and what each of them does:

- **-c | –config (cmd only):** This option accepts a string argument; the location of the configuration file.
- **-u | –g1 (cmd), network1 (cfg):** This option accepts a string argument; the first of the pair of networks we want to align.
- **-v | –g2 (cmd), network2 (cfg):** This option accepts a string argument; the second of the pair of networks we want to align.
- **-s | –s1 (cmd), sigs1 (cfg):** This option accepts a string argument; the spectral signature file for the first network.
- **-t | –s2 (cmd), sigs2 (cfg):** This option accepts a string argument; the spectral signature file for the second network.
- **-x | –blast (cmd), sequencescores (cfg):** This option accepts a string argument; the file containing the all-vs-all BLAST e-values which are used as sequence-based distances between the vertices of the two networks. This parameter is *optional*, and if it is left out, then the networks will be aligned using topology alone.

- **nneighbors (cfg only):** For each vertex in `network1`, the number of nearest neighbors in `network2` to which the distance should be computed. The value `all` denotes that for each vertex in `network1` (which contains, say, N vertices), the distance to the N closest vertices in `network2` should be computed. Numeric values other than `all` may be provided (e.g. a value of 100 would store the distance to only the 100 closest vertices). Providing a value other than `all` can reduce the memory usage, but it also artificially limits the search space, so we recommend against it when possible.

- **-l | –localSearch (cmd), searchiter (cfg):** This option accepts an interger argument; the number of local search iterations that should be performed after the initial global alignment is complete. Each search iteration loops over all aligned pairs and attempts to swap them with other aligned pairs that will increase the topological quality of the alignment without decreasing the sequence similarity of aligned pairs (subject to the "bad-move" ratio described below).

- **-b | –beta (cmd), beta (cfg):** This option accepts a numeric (floating point) argument. This numeric value represents a sequence distance (e.g. E-value) that is a hard threshold for aligning vertices across two networks. That is, if the sequence distance between vertex x from `network1` and vertex y from `network2` is greater than `beta`, then the algorithm will not allow these vertices to be aligned in the initial seed-and-extend phase of the algorithm.

- **-r | –unconstrained (cmd), ratio (cfg):** This option accepts a numeric (floating point) argument. This value is the ratio of "bad-moves" allowed during the local-search phase of the alignment algorihm. This is equivalent to the "budget" parameter described in (Patro and Kingsford 2012), and we refer the user to the paper for a more thorough explaination of this parameter.

- **-o | –output (cmd only):** This option accepts a string argument; the name of the file where the resuling alignment will be written (in the format described in Alignment File Format).

- **-a | –alpha (cmd), alpha (cfg):** This option accepts a numeric (floating point value) argument. This value is the number by which the spectral (i.e. topological) and sequence-based distances will be combined according to alpha * topo + (1 - alpha) * seq. If this option is not provided on the command line or in a configuration file, GHOST will attempt to automatically compute a *reasonable* value for alpha that puts the topological and sequence-based distanace onto a similar scale.

References

Patro, Rob, and Carl Kingsford. 2012. "Global network alignment using multiscale spectral signatures." *Bioinformatics* 28 (23): 3105–3114.