PNNL-22365

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Species Paralogy and Orthology Clique Solver (SPOCS) User Guide

SPOCS Version 1.0.10

April 2013

Summary

The Species Paralogy and Orthology Clique Solver (SPOCS) was written to provide robust software that implements a graph-based approach to ortholog and paralog prediction among a group of species. SPOCS accepts protein sequence files in FASTA (http://en.wikipedia.org/wiki/FASTA_format) file format, uses NCBI BLAST (http://en.wikipedia.org/wiki/BLAST) to identify reciprocal best hits, and a maximum clique algorithm to identify the orthologs and paralogs. SPOCS returns the predicted orthologs and paralogs in a tab-delimited report, and optionally, in a self-contained HTML output with visualizations of the ortholog relationships.

This guide is intended to instruct the user regarding building and running the SPOCS software from the source code on systems running Linux RHEL (http://en.wikipedia.org/wiki/Rhel) 5.x or Macintosh OSX operating systems, and provide information regarding the output formats.

SPOCS was developed by DS Curtis, AR Phillips and LA McCue with funding from the Genome Sciences Progam of the US DOE (http://genomicscience.energy.gov/program/index.shtml).

Contents

1.0 Introduction

The SPOCS application is designed to identify an orthologous group of proteins as a clique (http://en.wikipedia.org/wiki/Clique_problem) made up of pair-wise reciprocal best hits. We use the following commonly accepted terms to describe the various relationships between genes/proteins. Orthologs are homologous genes/proteins originating from a speciation event, whereas paralogs arise from a gene duplication event. Within a lineage, those paralogs arising from a duplication event that predates a speciation event are out-paralogs, and those arising from a duplication event subsequent to speciation are in-paralogs. SPOCS follows the conventions of InParanoid (Remm, M., Storm, C.E. and Sonnhammer, E.L. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons, *J Mol Biol*, **314**, 1041-1052) to identify orthologs and in-paralogs between two species, merges the pairwise ortholog/paralog relationships, and identifies cliques in each resulting graph by breaking it into subgraphs and using the branch and bound clique-finding algorithm (http://www.sicmm.org/~konc/maxclique/) to iteratively to find all cliques in each subgraph.

2.0 System Requirements

2.1 Hardware

The SPOCS application will run on 64-bit Intel based systems with dual quad core processors with a minimum of 8 GB of memory. If the data include sequences over 10,000 characters, it is recommended that the system have 16 GB of memory.

2.2 Operating System

The SPOCS application has been built and tested on Linux Red Hat Enterprise 5.2 and Macintosh OSX. The OS should be installed to run in 64-bit mode. SPOCS may build and run on other UNIX-based operating systems, but this has not been tested.

2.3 Software

SPOCS uses NCBI BLAST (http://www.ncbi.nlm.nih.gov/guide/data-software/) to do the initial processing of the FASTA protein data. You must have the C toolkit applications, i.e., blastall, in your PATH to build and run SPOCS. From your home directory, type the following command to see if BLAST is in your PATH:

```
which blastall
```

The full path will be returned if BLAST is in your PATH. SPOCS does not currently support BLAST+.

2.4 Required Tools/Libraries

A C++ compiler and the Boost C++ libraries are required to build SPOCS. **NOTE:** the same compiler must be used to build the Boost libraries and SPOCS.

MacOS X compiler:

a. The Xcode developer tools are required; install Xcode from the Apple Store.

b. Start Xcode, then go to the menu Xcode → Preferences → Downloads and install the "Command line tools".

Linux compiler:

a. The GNU g++ compiler, version 4.1.2 or newer, is required. From your home directory, you can check that g++ is in your PATH, and if so, the version:

```
which g++
g++ --version
```

b. You may need to set an environment variable for the location to ensure the correct compiler is used. For example, if your GNU g++ is in /opt/local/, set an environment variable using one of the following commands:

```
setenv CXX_PATH /opt/local (for csh shell)
export CXX_PATH=/opt/local (for bash or sh shell)
```

The Boost C++ libraries, version 1.33-0 or newer, from http://www.boost.org are required. The installation of these libraries should be done using the following steps to install the headers and libraries in `/usr/local/`:

MacOS X:

a. See http://www.macports.org and install MacPorts if it is not already installed

b. Run the following command: `sudo port install boost`

Linux:

a. If Boost is not already installed on your system, you can download the current version: http://sourceforge.net/projects/boost/files/boost/1.53.0/

b. There are many ways to install Boost on your system. An example shell script to build Boost from scratch in the same manner that we have it installed on our systems is included in the support directory within the SPOCS download. This shell script (boost_install_script.sh):

i. will use wget to download Boost version 1.53.0
ii. is designed to use the bash shell on Linux
iii. will build the Boost libraries in: BUILD_DIR, /var/tmp
iv. will install the libraries in: INSTALL_DIR, /usr/local
v. will build the 64-bit versions of the libraries
vi. will build "static" (LIB_MODE) versions of the libraries
vii. will build "release" (BOOST_VARIANT) versions of the libraries

Modify this shell script as needed for your system, and install boost.

2.5 Building SPOCS

The SPOCS download is a tarball of the files you will need to build SPOCS. **NOTE:** you need to use the same compiler to compile SPOCS as that used to compile the Boost libraries.

The following commands will install SPOCS:

```
tar zxf spocs.tar.gz
cd spocs
./configure --with-boost=/path/to/boost
make
make check
make install
```

This will install spocs in /usr/local/bin/. If the last step returns an error, you likely do not have permission to install in that dir. The installation directory can be changed using the `--prefix` option with the `./configure` command.

You should now be able to type the following command and get output similar to that shown:

```
/usr/local/bin/spocs -v

Species Paralogy and Orthology Clique Solver (spocs)

   spocs version:            1_0_10
   Compiled with:            GNU /usr/local/gcc_4_7_1/bin/g++ -v 4.8.0
     Optimize Level:         O2
     Debug:                  ON
     Boost library version:  1_53
   Compiled on:              Mar 19 2013, 15:27:17
   Report Bugs to:           leeann.mccue@pnnl.gov
```

3.0 Running SPOCS

3.1 Command Line Options

These options are allowed on the command line. The format of each option below is short/terse command line option OR (||) long/verbose command line option. For example, you can enter `spocs –h` or `spocs --help` to display the help information.

`-h || --help`
Print a help message on how to use spocs and exit. A minimal explanation of each option is shown and if an argument (denoted by the word `arg`) is required the default value is shown.

`-v || --version`
Print a version string listing when the executable was produced, the name and version of the compiler used, the compiler optimization settings, whether debug symbols are in the executable or not, the version of the Boost.org C++ libraries used, and the date and time the executable was created. After printing this information and the copyright information, the program exits.

`-s args || --species-list list_of_fasta_files`
Specify the list of species FASTA files. For convenience, the user can put all the FASTA files (including the `--outgroup-file` file) in a single directory and specify all the files with a wild card (e.g.: `*.fasta` or `*.faa` or `*.txt`). IMPORTANT: Each species must have a single fasta-format file and the gene/protein names (or ids) in the FASTA files must be unique across all

the files. If a name/id is duplicated in a file or between files, the spocs software will stop processing and fail with an error. *This is required*.

`-R arg || --report-dir=<arg>`

Specify an existing directory in which to write the text and HTML reports. *This is required*.

`-p arg || --paralog-output-dir=<arg>`

Specify an existing directory to put all of the generated BLAST scripts, the output of the BLAST runs, the hits file that combine overlapping alignments, and the resulting pairwise ortholog/paralog files (results comparable to that of the Inparanoid output). If the debug option is specified, the ortholog, journal, and rejected debugging information files are written to this directory as well. *This is required*.

`-o arg || --outgroup-file=<arg>`

Specify a FASTA file that will be used as an outgroup, i.e., a species that is more distantly related to your species of interest than any of the species of interest are to each other. The software will find reciprocal best BLAST hits between two species of interest (A and B), and will throw out any hits between A and B that score lower than the reciprocal best hits between A and C or B and C. NOTE: If the outgroup file is also included in the `--species-list`, it will be removed from the species list and handled correctly.

`-d || --debug`

Create the debugging information files for finding orthologs, general procedural (journal) steps, and reasons for rejecting orthologs and paralogs.

`-f || --force`

SPOCS tries to cache as much information and use existing output if it exists (e.g., existing BLAST output). If the user specifies the force option, all cached information is ignored and recreated.

`-g arg || --seg-overlap=<arg> (=0.25) (Range between 0.0 to 1.0)`

When parsing the BLAST results, if the alignment of two proteins is broken up into multiple segments, this is the percentage of the total length of each sequence that needs to be included when considering all the segments. The default is that both the A and B sequence must have at least 25% of its full length in the alignments. (Example: if a protein of length 100 has position 1-20 and 80-100 in local alignment segments, then the total length in the alignment segments is 40, or in this case 40% of the full length of the protein.)

`-q arg || --seq-overlap=<arg> (=0.5) (Range between 0.0 to 1.0)`

When parsing the BLAST results, the combined alignment (after combining aligned segments) must include at least this percentage of the length of each sequence. The default is that both the A and B sequence must have at least 50% of its full length in the alignment. (Example: if a protein of length 100 has position 1-20 and 80-100 in local alignment segments, when those are combined the alignment stretches from position 1 to 100, covering 100% of the protein.)

`-c arg || --outgroup-cutoff=<arg> (=50) (blastp bit score)`

BLAST hits to the outgroup species must have a bit score greater than or equal to this value.

`-b arg || --blast-score-cutoff=<arg> (=50) (blastp bit score)`

BLAST hits between two species of interest (species A and species B) must have a bit score greater than or equal to this value to be retained.

`-e arg || --evalue=<arg> (=10) (blastp evalue)`

Only report BLAST hits that have an E-value score less than or equal to this value. Altering this value could eliminate some BLAST hits with low bit scores and will change the amount of BLAST data saved in the directory specified with the –p option. NOTE: The default value, 10, is the same as in the BLAST software.

`-m arg || --matrix=<arg> (=BLOSUM62)`

The scoring matrix used by blastp. This can be changed to any matrix stored in your blast/data/ directory.

`-i arg || --inner-confidence=<arg> (=0.05) (Range between 0.0 to 1.0)`

The minimum confidence value for retaining in-paralogs in an ortholog group. NOTE: This is as described in the InParanoid documentation.

`-r arg || --group-overlap=<arg> (=0.5) (Range between 0.0 to 1.0)`

Merge separate ortholog groups if a protein in one ortholog group has >= this confidence value as an in-paralog in another ortholog group. NOTE: This is as described in the InParanoid documentation.

`-C arg || --confidence=<arg> (=1.0) (Range between 0.0 to 1.0)`

The confidence value for retaining orthologs and in-paralogs between each pair of species. NOTE: This confidence is calculated as described in the InParanoid documentation.

`-w || --write-scripts-only`

This option preprocesses all the data files and creates all the scripts that need to be executed as BLAST runs, then exits instead of running them. For example, if you have species A and species B but no outgroup, this option would validate the input files then produce four scripts called A-B.sh, B-A.sh, A-A.sh and B-B.sh. The scripts can then be run on separate systems or processors in parallel to run BLAST and produce the paralogs output files (the files containing the InParanoid ortholog and in-paralog identifications). Once the paralogs files exist in the directory specified with the –p option, you can run the same spocs command without the `-w` option to finish solving the graphs and finding the cliques.

`-H || --html-cliques`

This option produces a set of related HTML pages that allow the user to interact with the graphs, data, and metadata. The HTML pages can be displayed in Chrome or Firefox or Safari as a standalone mini-application front-end to the graph, clique, and optional metadata. If this option is not supplied on the command line, orthologs are only provided in the ASCII file: cliques.report.

`-T arg || --type=<arg> (=CSID)`

The type of graphs that you want spocs to produce. The argument can be any combination of the letters C, S, I, and D to indicate Complete cliques, Semi-Complete cliques, Incomplete graphs with multiple cliques, and Degenerate graphs with multiple cliques.

`-N arg || --max-nodes=<arg> (=2000)`

This is the maximum number of nodes per subgraph to process. The default value of 2000 is appropriate on a system with 8 GB of RAM; the value can be higher on a system with more memory. This parameter keeps the clique solving problem from becoming too large (NP Complete).

`-M arg || --min-clique-size=<arg> (=2)`

The minimum number of nodes (proteins) in a clique.

`-B arg || --bad-paralog-cutoff=<arg> (=5)`

The maximum number of nodes from the same species in a single ortholog/in-paralog group. Use the value 0 to disable this cutoff. Proteins removed from cliques because of this option are reported in the cliques.report.rejected output file (see below).

`-E arg || --edge-threshold=<arg> (=0.05)`

The percentage of the maximum possible number of edges in a complete graph that may be missing to call it a Semi-Complete clique. For example, in a graph with 8 nodes (proteins), the maximum possible number of edges is 28; in this case, to be called a Semi-Complete clique only 1 edge could be missing (int (0.05 * 28) = 1).

`-D arg || --metadata-file=<arg>`

Specify an existing file in CSV format containing gene/protein expression data that will be used to color the graph in the visualization portion of the HTML output.

`-V || --validate-metadata-only`

Validate the metadata file content and structure, then exit. This is a preprocessing step to skip all the expensive processing if the metadata is invalid.

3.2 Running SPOCS

The SPOCS download includes a `data/` subdirectory containing example data and output to test your installation of SPOCS. These data are used during local installation (see section 2.5): the “`make check`” command will test the installation and generate SPOCS output files.

The test data consists of the protein fasta files for 4 intensively studied, closely related gamma-proteobacteria: *Escherichia coli* str. K-12 substr. MG1655, *Salmonella enterica* subsp. *enterica* serovar Typhimurium str. LT2, *Salmonella enterica* subsp. *enterica* serovar Typhi str. CT18, *Yersinia pestis* KIM10+, and the fasta file for an archaeal species (*Methanothermobacter thermautotrophicus* str. Delta H) to use as an outgroup. Also included is a small example file of metadata containing some expression values extracted from the literature.

To run SPOCS yourself using the test data, create a directory for the clique output (the `–R` option) and a directory for the BLAST and pairwise ortholog/in-paralog output (the `–p` option) and run the following command (modifying paths and names as needed) to generate output for these test data:

```
/path_to/spocs -s data/output/gold4_fasta/*.fasta
   –o data/output/gold4_fasta/OUTGROUP*.fasta
   -R data/mytest_gold4_cliques/ -p data/mytest_out_dir/ -H –M 3
```

Depending on your system, this may take an hour or two, mostly time required for the BLAST runs. The output file called “`cliques.report`” should be identical to the one included with the download and found in: `data/ref/gold4_cliques.report` (the format of this file is described in section 3.4).

If you continue to use the same out_dir for future SPOCS runs, SPOCS will always check to see if specific BLAST runs already exist, and will not redo these. For example, if after running the above command, you add a fifth and sixth species to the set of four provided in our test data and rerun SPOCS using the same out_dir, SPOCS will not redo the BLAST runs that have already been performed (provided you have left those results in the out_dir).

Input data requirements:

- All of the protein sequences for each species must be present in a single fasta file.

- The protein sequence identifiers (the text strings immediately following the '>' symbols in the fasta files) must all be unique among the complete set of all species and all proteins.
- The optional metadata file must be in CSV format, where rows are the fasta sequence ID's, and columns are metadata values for the given ID. The format is as follows:

  Header line:
  protein,[column1_name]:[min]:[max]:[a,r],[column2_name]:[min]:[max]:[a,r], ...
  ([a,r] specifies whether the column represents a ratio (min/max centered around zero) or abundance (allows arbitrary min/max) values)

  Data line:
  [fasta_id],[column1_value],[column2_value], ...

3.3 Ortholog output

Provided a set of *n* fasta files for *n* species), SPOCS proceeds through three main stages to identify orthologs. First, SPOCS executes a series of BLAST runs between every pair of species to identify reciprocal best hits; this is the most compute intensive stage, requiring n^2 independent BLAST runs (more if an outgroup is used). In the second stage, SPOCS uses the BLAST results to generate an orthology/paralogy relationship graph based on merging the pairwise ortholog/in-paralog relationships. The graphs are identified as one of four subjective types: Complete, SemiComplete, Incomplete, and Degenerate:

i. Complete graphs include no more than one protein per species, and every protein (node in the graph) is connected to every other protein by a reciprocal best hit (edge), i.e., a maximum clique.
ii. SemiComplete graphs are those in which no more than one protein per species is present but with less than the maximal number of edges (this is a tunable parameter for which the default is 5%).

   NOTE: By definition, the Complete and SemiComplete graphs are the result of single unique graphs that resolve to single unique cliques.

iii. Incomplete graphs have no more than one protein per species, and less than the maximal number of edges (more missing edges than allowed for SemiComplete).
iv. Degenerate graphs have more than one protein (node) from one or more of the species.

The last stage of SPOCS involves identifying as many maximum cliques as possible in the Incomplete and Degenerate graphs; cliques in each graph are identified by breaking the graph into subgraphs and using the branch and bound clique-finding algorithm to iteratively to find all cliques in each subgraph.

3.4 The text output: cliques.report

The ortholog predictions will be in the directory you specify with the `–R` option. A file named "cliques.report" will contain a simple tab-delimited table of orthologs, in which each line represents a clique of predicted orthologs. The first five columns of this table are:

1 – an ID number for the ortholog/paralog graph
2 – an ID number for the clique within that graph
3 – the type of clique: Complete, SemiComplete, IncompleteSolved, or DegenerateSolved
4 – the number of nodes (proteins) in the ortholog clique
5 – the number of edges (reciprocal best hits) in the ortholog clique
6 – the remaining columns have the ortholog data

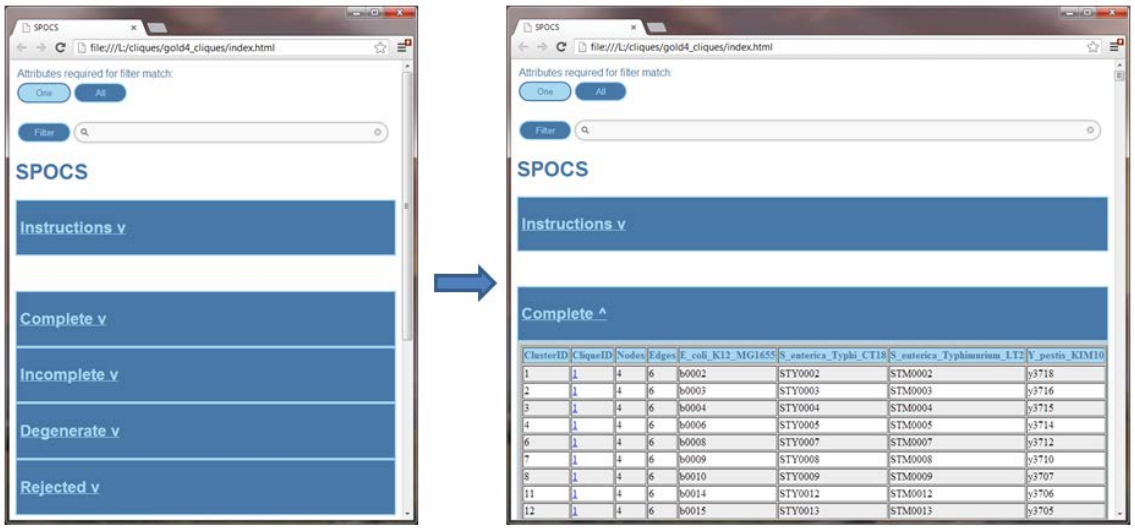
ClusterID	CliqueID	CliqueType	Nodes	Edges	E_coli_K12 _MG1655	S_enterica_ Typhi_CT18	S_enterica_ Typhimurium_ LT2	Y_pestis _KIM10
52	1	C	4	6	b0059	STY0111	STM0096	y3656
53	1	C	4	6	b0060	STY0112	STM0097	y3655
54	1	DS	3	3	b3583	STY4119	STM3677	-
54	2	DS	3	3	b0061	STY0118	STM0101	-

The column headers are the names of the fasta files minus the file extension. The protein IDs are the first string following the “>” symbol for each sequence in the fasta file. In the above excerpt from a cliques report, the first two ortholog groups are Complete cliques. The second two ortholog groups are two cliques (CliqueID 1 and 2) identified from a single Degenerate graph (ClusterID 54) that contained two proteins from each of the first three species.

SPOCS also generates a file called “cliques.report.rejected”. This file simply lists the protein IDs of proteins that were identified during the initial pairwise ortholog prediction stage, as exceeding the maximum number of proteins from the same species that belong in a single ortholog/in-paralog group. This maximum is set to 5 by default. This instance is triggered when a single species encodes several proteins that are identical or nearly identical; these proteins are identified with high confidence as a many-to-many ortholog/in-paralog group. We have observed this in species that have many copies of a transposon or insertion element, resulting in many copies of an identical transposase protein. As this is generally not of interest in ortholog prediction, we provide a mechanism (the `–B` parameter) to limit reporting these cases in the ortholog table, but then do report these proteins in the “cliques.report.rejected” file.

3.5 The HTML output: index.html

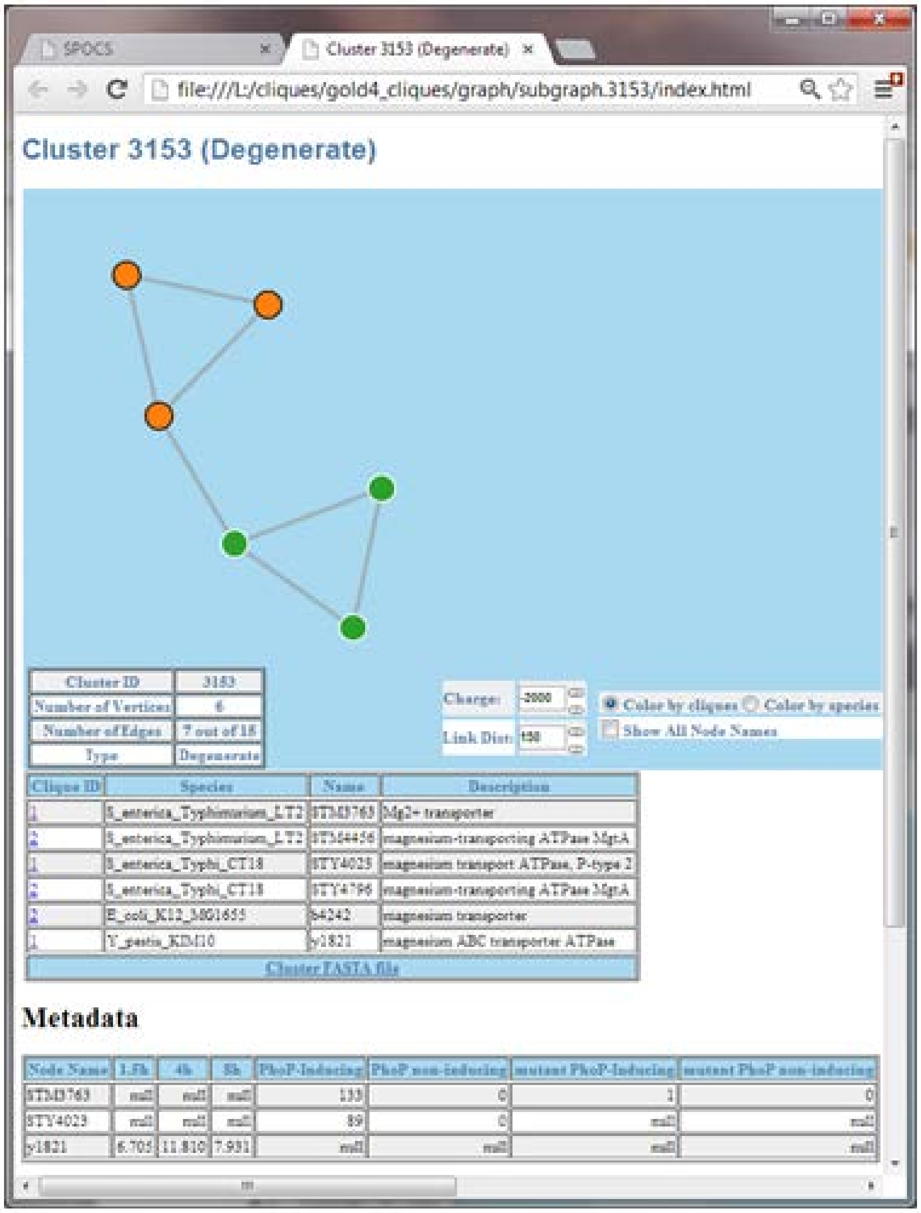
If the `–H` option is used, self-contained HTML output of the ortholog predictions will also be generated and will appear in the directory you specify with the `–R` option. Use a Chrome or Firefox or Safari browser window (NOTE: InternetExplorer is not supported) to open the index.html file and the start page will appear, from which the list of ortholog cliques in each of the clique categories present in your data can be expanded:


SPOCS
file:///L:/cliques/gold4_cliques/index.html
Attributes required for filter match:
One
All
Filter
SPOCS
Instructions v
Complete v
Incomplete v
Degenerate v
Rejected v
SPOCS
file:///L:/cliques/gold4_cliques/index.html
Attributes required for filter match:
One
All
Filter
SPOCS
Instructions v
Complete ^

ClusterID	CliqueID	Nodes	Edges	E_coli_K12_MG1655	S_enterica_Typhi_CT18	S_enterica_Typhimurium_LT2	Y_pestis_KIM10
1	1	4	6	b0002	STY0002	STM0002	y3718
2	1	4	6	b0003	STY0003	STM0003	y3716
3	1	4	6	b0004	STY0004	STM0004	y3715
4	1	4	6	b0006	STY0005	STM0005	y3714
6	1	4	6	b0008	STY0007	STM0007	y3712
7	1	4	6	b0009	STY0008	STM0008	y3710
8	1	4	6	b0010	STY0009	STM0009	y3707
11	1	4	6	b0014	STY0012	STM0012	y3706
12	1	4	6	b0015	STY0013	STM0013	y3705

The columns in the HTML table are the same as those described above in the “cliques.report” file. Here, the CliqueID is a clickable link that will open a new tab with a visualization of that clique. For the Incomplete and Degenerate categories, both the ClusterID and the CliqueID are clickable links.

For example, clicking on a ClusterIDs for a Degenerate graph a new tab appears:


SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/graph/subgraph.3153/index.html

Cluster 3153 (Degenerate)

Cluster ID	3153
Number of Vertices	6
Number of Edges	7 out of 18
Type	Degenerate

Charge: -2000
Link Dist: 150
(● Color by cliques ○ Color by species)
☐ Show All Node Names

Clique ID	Species	Name	Description
1	S_enterica_Typhimurium_LT2	STM3763	Mg2+ transporter
2	S_enterica_Typhimurium_LT2	STM4456	magnesium-transporting ATPase MgtA
1	S_enterica_Typhi_CT18	STY4023	magnesium transport ATPase, P-type 2
2	S_enterica_Typhi_CT18	STY4796	magnesium-transporting ATPase MgtA
2	E_coli_K12_MG1655	b4242	magnesium transporter
1	Y_pestis_KIM10	y1821	magnesium ABC transporter ATPase
Cluster FASTA file			

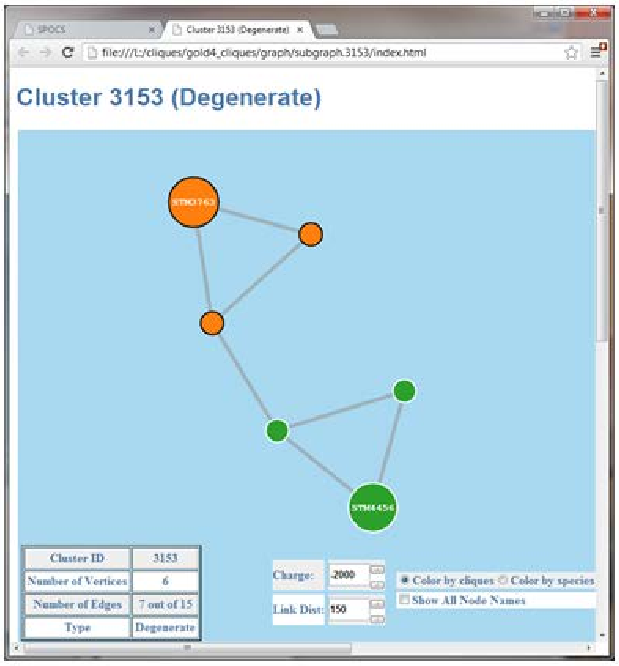
Metadata

Node Name	1.5h	4h	8h	PhoP-Inducing	PhoP non-inducing	mutant PhoP-Inducing	mutant PhoP non-inducing
STM3763	null	null	null	135	0	1	0
STY4023	null	null	null	89	0	null	null
y1821	6.705	11.810	7.931	null	null	null	null

The separate cliques identified from this Degenerate graph are shown in different colors (orange and green nodes), and clicking either a particular node or the number of one of the CliqueIDs will open another tab for just that ortholog clique. The table below the graph provides the protein IDs and a description extracted from the fasta file (the text following the protein ID). The fasta sequences for this ortholog graph are available by clicking on the "Cluster FASTA file" link at the bottom of this table. If metadata on the gene or protein expression of these genes has been provided at run-time, then those protein nodes will be outlined in black (see the orange nodes in the figure), and the metadata are also displayed in a Metadata table below the protein ID table.

3.5.1 Interacting with the graphs

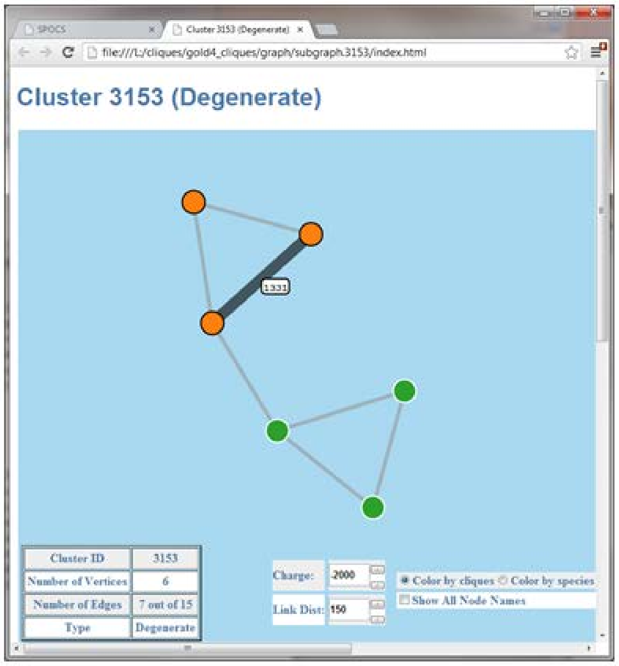
The graph image is very interactive. Its structure is computed using a force-directed layout engine, which uses charged particles to place related genes closer together in the graph. Thus, the user is able to modify the particle charge and minimum link distance using controls on the web page (this is available for Incomplete and Degenerate graphs, not cliques). The graph can also be repositioned by clicking on a node and dragging it to a desired location. Hovering over a node will enlarge that node and any other nodes from the same species, and display the protein ID (left image), and hovering over an edge will display the BLAST bit score between those two proteins (right image):


SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/graph/subgraph.3153/index.html
STM3763
STM4456

Cluster 3153 (Degenerate)

Cluster ID	3153
Number of Vertices	6
Number of Edges	7 out of 15
Type	Degenerate

Charge: -2000
Link Dist: 150
[✓ Color by cliques] Color by species
Show All Node Names


SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/graph/subgraph.3153/index.html
1331

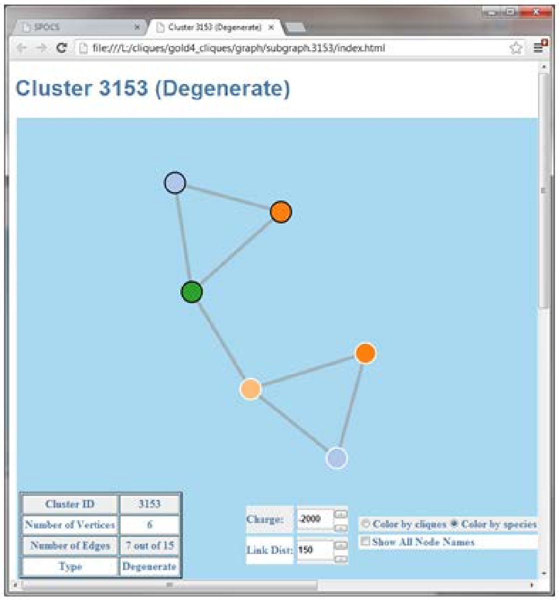
Cluster 3153 (Degenerate)

Cluster ID	3153
Number of Vertices	6
Number of Edges	7 out of 15
Type	Degenerate

Charge: -2000
Link Dist: 150
[✓ Color by cliques] Color by species
Show All Node Names

3.5.2 Coloring the graphs

The graphs are colored “by clique” by default, thus when the above Cluster 3153 window was opened, the two Cliques identified by the branch and bound clique-finding algorithm appear in different colors (orange and green). However, the user has a choice of the following additional node color options. (1) A radio button allows you to view the graphs “by species” such that each protein node from the same species is given the same color (below, left image). (2) A radio button allows you to view the graphs with coloring according to the metadata. In this mode, nodes for which no metadata are available appear gray, nodes for which the metadata file had a NULL (empty) value appear white, nodes for which raw abundance data are available are given a unique color according to the metadata column, and nodes for which expression ratio data are available appear with red-to-green (positive-to-negative) shading and hovering over a node (below, right image) will show the range of red-to-green (limits are extracted from the metadata file).


SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/graph/subgraph.3153/index.html

Cluster 3153 (Degenerate)

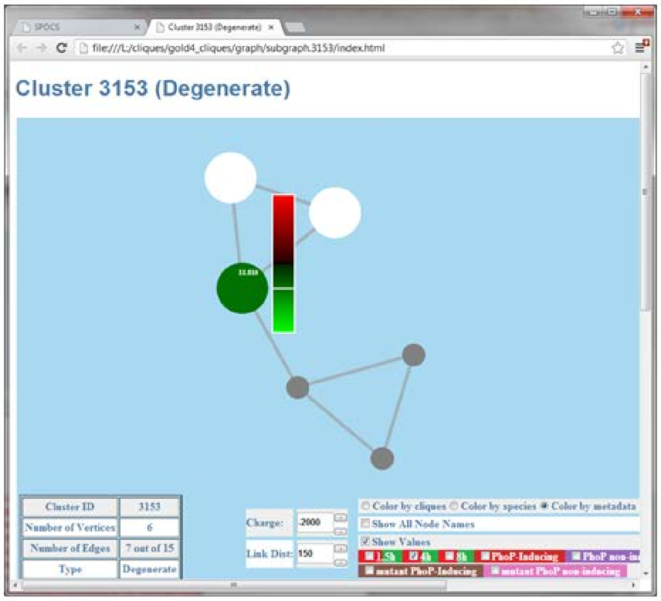
Cluster ID	3153
Number of Vertices	6
Number of Edges	7 out of 15
Type	Degenerate

Charge: -2000

Link Dist: 150

Color by cliques [✓ Color by species]

☐ Show All Node Names


SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/graph/subgraph.3153/index.html

Cluster 3153 (Degenerate)

Cluster ID	3153
Number of Vertices	6
Number of Edges	7 out of 15
Type	Degenerate

Charge: -2000

Link Dist: 150

Color by cliques Color by species [✓ Color by metadata]

☐ Show All Node Names

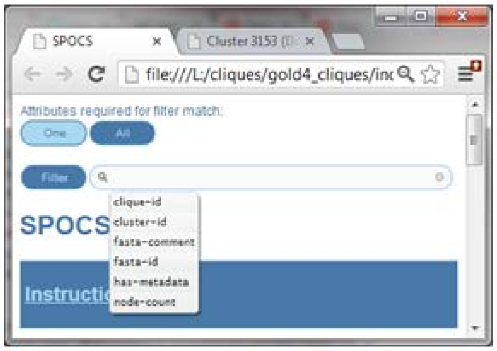
☑ Show Values

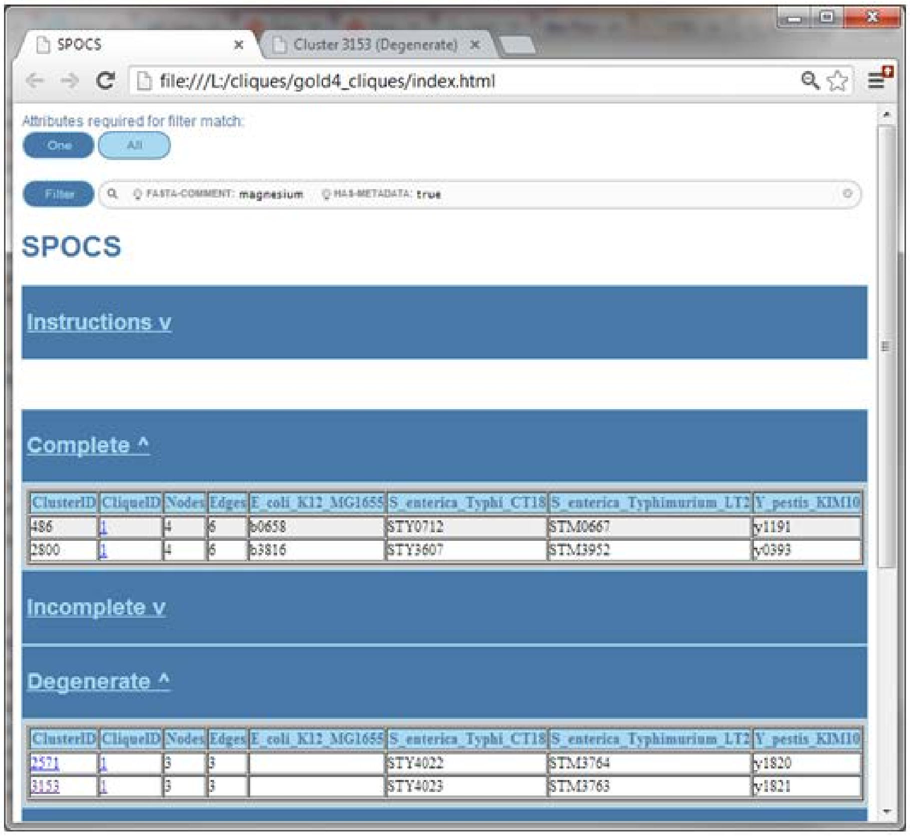
☐ 1.5h ☑ 4h ☐ 8h ☐ PhoP-Inducing ☐ PhoP non-in

☐ mutant PhoP-Inducing ☐ mutant PhoP non-inducing

3.5.3 Searching the data

The data in SPOCS can be searched and filtered for attributes of interest. On the index.html start page, clicking in the open bar at the top next to the magnifying glass will show a selection of attributes. Selecting one then allows you to type in text to specify the search/filter parameter. In the example below, we have filtered on “magnesium” in the “fasta-comment” field (the text following the sequence identifier that typically contains gene function annotation information) and “true” for the “has-metadata” field. Clicking “All” at the top of the page filters on all the parameters specified, then clicking “Filter” will filter the data visible in the table below. Each section must be expanded to view the results (by default they will remain collapsed until expanded by the user).




SPOCS
Cluster 3153 (Degenerate)
file:///L:/cliques/gold4_cliques/index.html
Attributes required for filter match:
One
All
Filter
FASTA-COMMENT: magnesium
HAS-METADATA: true
SPOCS
Instructions v

Complete ^

ClusterID	CliqueID	Nodes	Edges	E_coli_K12_MG1655	S_enterica_Typhi_CT18	S_enterica_Typhimurium_LT2	Y_pestis_KIM10
486	1	4	6	b0658	STY0712	STM0667	y1191
2800	1	4	6	b3816	STY3607	STM3952	y0393

Incomplete v

Degenerate ^

ClusterID	CliqueID	Nodes	Edges	E_coli_K12_MG1655	S_enterica_Typhi_CT18	S_enterica_Typhimurium_LT2	Y_pestis_KIM10
2571	1	3	3		STY4022	STM3764	y1820
3153	1	3	3		STY4023	STM3763	y1821

4.0 System Resources

The SPOCS software can consume as much memory and CPU resources as you have available. It is recommended that you do some testing with smaller data sets and monitor how the system performs before adding more species.

5.0 Output Messages

All messages consist of fatal conditions. All messages have the following format:

```
 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LEVEL
  File:      FILENAME.EXT(line:###)
  Class:     CLASSNAME
  Function:  FUNCTION
  Message:   Error Code=CODE
             CONTEXT (what was the code trying to do)
             ACTUAL MESSAGE.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Where:

`LEVEL` is `CAUTION`, `ERROR`, or `FATAL`.

`FILENAME.EXT` is the name of the source code file where the message originated.

`###` is the line number in the `FILENAME.EXT` where the message originated.

`CLASSNAME` is the name of the object-oriented class where the function exists.

`FUNCTION` is the name of the function where the message originated.

`CODE` is the condition associated with the message. For instance, the `CODE` can be `BadOption` if a value is invalid.

`CONTEXT` explains what the code was trying to do. For example, a context for parsing command line options would be “Processing command line options”.

`MESSAGE` explains the condition. For example, “unknown options -j”.

6.0 Troubleshooting

If you are getting data that does not make sense or unexpected results, try removing all the files in your output directory (specified by –p command line option) and all the files in the output report directory (specified by the –R command line option), and try to run the code again.

7.0 Reporting Bugs

Please submit bug reports to the e-mail address in the output of `spocs –v` and include as much information as possible to describe the problem. Please include the output from the commands in the “Troubleshooting” section and the output from `spocs –v` up to the copyright message.