SUPER3GL USER'S MANUAL (V.1.4.5)

Table of Contents

1. Purpose

The program `super3GL` is intended for solving a typical bioinformatic problem in the field of phylogenetics - construction of a so-called *supertree*, i.e. a tree that is the closest, in the certain sense, to the given set of trees. The source data is the set of trees which reflect evolution of organisms or their biological attributes for some taxonomy. The result is a supertree, which leaves correspond to taxa of the same or higher level of detail, so that it agrees with the set of source trees as good as possible in sense of a predefined criterion. For example, on the basis of a set of gene trees a supertree can be constructed of species containing these genes which has the minimal cost with regard to the source set (for given costs of a gene loss and/or gene duplication evolutionary events). A necessary condition is that for each leaf of a source tree (i.e. for each gene in this example) it is known to what leaf of a supertree (i.e. a species) it belongs. This information is represented directly in a set of source trees with use of special syntax (see 4.1). Notions "gene" and "species" are not necessarily identical to same biological terms, and are used in this manual only for the sake of definiteness.

2. Algorithm

The program `super3GL` implements the algorithm described in **[1]**, which has some improvements such as parallel processing capability. The algorithm consists of two phases: (1) construction of a set of basis trees, (2) incremental construction of a supertree by induction. The first phase should be executed entirely in one run of the program and cannot be interrupted; its results can be saved as a file of basis trees.

The second phase is carried out either directly after the first phase or in a separate run of the program. The processing can be interrupted at any moment and resumed later, starting from the last unfinished induction step. Result of each induction step (including final one) is the current constructed supertree. Running of the program for start or continuation of the phase 2 processing is referred to hereinafter as *resume mode*.

The source gene trees may be not only rooted binary trees, but also include polytomous nodes with any number of descendants. However, the correct rooting of the tree is assumed.

3. Implementation and building

The program `super3GL` is written in C++ and has a command-line interface. After proper recompilation, it can work under any 32- and 64-bit operating systems Windows, Linux, Unix, and MacOS. The program detects presence of the multiprocessor environment corresponding to the standard MPI version 1.2 or later [2], and in this case it automatically involves the parallel processing capabilities implemented in the algorithm. If a high-performance cluster is available, this allows reducing considerably the solution time for sizeable tasks (see the program performance data in Section 7). In some cases, where a large memory is available, the program performance can also be increased by changing of the working data structure (a sort of time/memory trade-off); in such situation a 64-bit version of the program should be used to work normally with the memory volume of 2–4 GB and more.

The following download options are available at the program web page [5]:

- binary executable module for Windows 32 bit (without parallelization)
- binary executable module for Windows 64 bit (without parallelization)
- source code to build the program in Linux (with or without parallelization) is provided under the GNU General Public License version 3 (or above).

The program building in Linux is accomplished by the standard utility `make`. The `makefile` provided in the distribution has been prepared to build a uniprocessor binary (without parallelization), but the user may build a parallel version of the program if an implementation of MPI v1.2 (or above) is available in the system. The user should edit the `makefile` appropriately to obtain an MPI-enabled binary on the specific system (two typical variants of possible changes are already commented in the distributed `makefile`).

A user can check downloaded or built binary by running it with `-h` option (the program will output general information about it including the version number and help on command line arguments, see 6.2 for detail).

The program options (parameters and source data) can be specified in the configuration file (Section 6.1) and/or in the command line when running the program (Section 6.2); in the latter case they are used in preference to options specified in the configuration file. In particular, the program can be run with no parameters at all, provided appropriate options are specified in the of configuration file or set to default values. So, in the uniprocessor environment a general form of a command line to run the program can be the following:

```
super3GL [options]
```

In the MPI environment a typical command line to run the program on 128 processors for 90 minutes can be:

```
      mpirun -np 128 -maxtime 90 super3GL [options]
```

(the exact syntax depends on specific realization of MPI-environment and is described in its documentation). Optional parameters of the program are shown here in brackets.

Note: While the command line in Windows is case-insensitive, other operating systems such as Linux can depend on the case of characters in a command line, so we recommend following the capitalization shown in examples throughout this manual including the program name.

4. Input data

When operating in *normal* mode (i.e. starting from the Phase 1), the program uses a set of gene trees as input data. The set consists of at least one file of input trees (Section 4.1). Each file, in its turn, contains at least one tree in Newick parenthesis format **[3, 4]**. The source tree(s) can also be provided directly in the `super3GL` configuration file to avoid using separate input tree file(s).

When operating in *resume* mode, the program does not use gene tree file(s) and a table species (see below). Instead, an intermediate file of basis trees (Section 5.3) built at Phase 1 is used. In addition, if a current state of supertree file (Section 5.1) is available, the supertree building will be continued from that state; otherwise, the supertree will be constructed from the first step of induction in the Phase 2.

As a complement to the file of input or basis trees, the table of species *can* be used in both modes provided as separate file (Section 4.2). This table serves the triple purpose: a) it gives a meaning of abbreviations which are frequently used for high taxa ("species"); b) it determines the maximum number of species involved in a problem; c) it allows the program to check input trees for correct species names. If the species table is not provided, the program does not check input trees, the maximum number of species needs to be specified, and the set of species is completed on the basis of input trees. In any case, `super3GL` works with those species names which are specified at leaves of input trees, and gives out results in the same notation. If desired, longer names of species from the table can be substituted for abbreviations in any tree with use of a stand-alone utility `uncode` available at the program web page **[5]**.

Nominally, input data include the configuration file (Section 6.1) in which the user sets the operating mode, parameters of the program, location of the input file(s), etc. In view of special importance, this and other information about the program management is gathered into separate Section 6.

4.1 File of input trees

The name of the input tree file is specified by the command line option **`-t`** and/or the parameter **`GeneTreeName`** in the configuration file. The name may include a path complying with rules of the operating system, but wildcards are not allowed. These parameters may appear more than once in the command line or configuration file; in such cases the set of input trees comprises all trees from all input tree files specified. The command line option **`--t`** cancels inclusion of *all* trees specified in the configuration file. The program checks only format of input trees, while the user is responsible for the consistency and relevancy of those trees.

The input tree file is a usual text file containing one or more trees in Newick format **[3, 4]**, each ending with a semicolon. Whole tree can occupy one line of the file or be wrapped to the subsequent lines according to the format rules. If the file contains more than one tree, each tree has to start in a new line. A tree can include optional comments, edge lengths, and inner node labels – all in accordance with Newick specification.

The only special feature of the input trees is a leaf label format. It is assumed that all leaves are labeled in the form

```
segm1[_segm2...][stop][...]
```

In other words, the leaf label consists of one or more segments separated by underscore characters and followed by optional stop character, which indicates that the remaining part of the label should be ignored. By default, the stop character is not used, but the user may specify one or several characters with **`StopLabel`** parameter in the configuration file, e.g. StopLabel = ”@*”. A segment itself cannot contain underscores, blanks and other characters not allowed in a label of the tree node according to Newick format.

It is also assumed that in each leaf label one or more initial segments in aggregate define the name of a high taxon (a “species”) from those to appear in the sough-for supertree. The exact number of such segments is the program parameter. Other segments of a label can indicate lower taxa, the name of an organism, a macromolecule, etc. (a “gene” in general); this optional information is not used in the current program version.

Theoretically, all leaves of all trees should have the same format of labels, and those labels must be unique within a tree. This is not always true in practice. Therefore, there is a program option **`StrictLabelControl`**, which determines a “strictness” of the tree format check. If the strict control is on, the program checks that the required number of initial segments does present in each leaf label, and entire labels are unique within a tree. If the strict control is off, source trees may contain labels in different formats, in particular, not including a gene name at all. In such situation, the program itself ensures the label uniqueness. In addition, the number of initial segments parameter becomes non-obligatory (a “species” name with smaller number of segments is not considered an error).

The program web page **[5]** provides examples of input tree files for both cases mentioned. Recall that the program allows for using several files in one task and the set of source trees consists of the union of the trees from all those files. However, the exact file names must be specified not using metacharacters * or ?. Specifying a directory name instead of all files it contains is not supported; the user should merge those files in a single input tree file.

4.2 Table of species

The species table is a character-separated text file. The character, which is used as delimiter, is the program parameter. There may be several tables of species, each one using its own delimiter. The species table(s) is/are specified by the command line option **`-s`** and/or by the parameter в **`SpeciesTableName`** in the configuration file. The name may include a path complying with rules of the operating system, but wildcards are not allowed. These parameters may appear more than once in the command line or configuration file; in such cases the union of species from all tables is used (however, the user is responsible for uniqueness of abbreviations or short names of species over all tables). The command line option **`--s`** cancels inclusion of *all* species tables mentioned in the configuration file.

It is not required to use the species table(s), but it can help to avoid many errors and to make the program results more readable with use of the stand-alone utility `uncode`, which can be downloaded for free from the program web page **[5]**.

The species table file may include several lines in the very beginning that contain field headings, descriptions and similar information, which is skipped by the program (the number of such lines in the species table is specified in the program parameter **`SkipLines`**). Subsequent lines must contain at least two fields separated by the delimiter character. The first field is an abbreviated notation of a species, and the second field is a full (or just readable) name of that species. `super3GL` ignores the second field, but short names in the first field must be unique over all species tables being processed in the program run. Species that are not involved in the source trees are silently ignored.

The species table file can be produced in e.g. Excel using data export in `.CSV` format. The program web page **[5]** provides an example of the table for Bacteria species.

5. Output data

The main result of the `super3GL` program execution is a supertree file (Section 5.1). As mentioned above, this file in its current state can serve an input data, when the program is run in resume mode.

In addition, a log file (Section 5.2) can be produced which: a) contains a copy of the information printed on the console (except of intermediate timestamps); b) contains auxiliary data describing the solution process, if requested by the user through the configuration file (Section 6.1).

When running in normal mode, the program can produce a basis tree file (Section 5.3) upon completion of the Phase 1. This file can be deemed output in such case, though it acts as input data for the program in resume mode.

5.1 Supertree file

The name of the supertree file can be specified by the command line option **`-o`** or by the parameter **`Super3Name`** in the configuration file (with preference of the command line). The name may include a path complying with rules of the operating system. If the supertree file is not specified, result tree is output to the console (`stdout` stream).

It is a text file that contains a current state of the supertree construction represented in Newick format **[3, 4]**. The supertree is always written in one line. During the Phase 2 of the algorithm, species are inserted in the tree by induction so the supertree is gradually growing. When the program finishes, this file contains the complete supertree built by the algorithm.

By default, the supertree file contains only a supertree in the parenthesis format. In addition, the program has an option to append, at each induction step, the current state of the supertree in the end of this file instead of completely rewriting it. Therefore, at each moment the supertree file holds the whole history of the supertree construction if this option is selected. The configuration parameter **`Super3Sequence`** is provided to control the choice.

At the minimum, a parenthesis notation of the supertree includes labels of all its leaves (“species” names). Depending on configuration parameters, the supertree can also include labels of inner nodes and lengths of edges. The length of the edge connecting a node with its parent is a number in between 0 and 1, which characterizes the “validity” of that node (the greater number, the more reliable node). In rare cases, when the validity cannot be calculated due to *x*/0 indeterminacy, a special value of –1 (or other value specified by the configuration parameter **`Uncertainty`**) appears to indicate that fact. For better display of the tree by other programs displaying scaled edges, a small positive value of this parameter can be a good choice. Similarly, zero value of the validity can often be better presented by a small positive number with use of the configuration parameter **`ZeroValue`**.

If requested in the configuration file (parameter **Super3Quality**), each line containing a tree in parenthesis notation in the supertree file is prepended with the following Newick comment:

```
ttt N=xxx (yyy added) TotalQ=zzz Q=vvv R=uuu [(ww species skipped)]
```

where `ttt` is elapsed time in minutes; `xxx` is a number of leaves (i.e. species) in the supertree; `yyy` is a short name (abbreviation) of a species inserted in the supertree at the last induction step; `zzz` is a quality of current tree (to be maximized by the program), and `vvv` is a quality of the inserted species; `uuu` is a reliability of the last species insertion that is expressed by a number in between 0 and 1 (the greater number, the more reliability, but the scale is essentially nonlinear). The record `ww...` , which can appear only if **SkipAmbiguos** parameter is true, indicates that `ww` species were rejected at this step of insertion as having zero reliability.

Examples of the supertree files are available at the program web page [5].

5.2 Log file

At user's option, the program `super3GL` creates a text file for logging its work. The file name (including optional path) can be specified by the command line option **-g** , otherwise, the log file named `super3GL.log` will be created in the current working directory. The option **--g** allows the user to cancel logging. If the file with such name already exists, then in normal mode the program deletes all existing data, while in resume mode it appends new data to the end of existing log file.

By default, the log file contains a copy of all information the program outputs to console (`stderr` stream) including error messages. It can be helpful at many clusters where the user has no access to the system console. To save the log file size, it does not contain service stamps (see **Milestones**) regularly displayed one over another by the program to confirm its work.

Examples of the log files are available at the program web page [5]. Information in the log file is mostly self-explanatory. The output starts with several lines informing about the program name, version, authors and licensing.

If any command line options are specified when running the program, then the line `Arguments: ...` is printed, where all those options are reproduced instead of ellipsis.

The next line in the log file (`Options: ...`) informs about main parameters and working modes of the program that are used in this computation. The full list of possible options in this line follows (not all elements necessarily appear in every situation):

`Resume` – means that the program is running in resume mode. This mode can be switched on by the command line option **-r** or the configuration parameter **Resume**. If command line option **--r** is specified, the resume mode will be switched off in spite of a setting in the configuration file.

`Prune(p=xxx)` – means that the program carries out the pruning of source trees by removal of species which rarely (or never) occur in the set of source gene trees. The occurrence threshold *p* is shown in parentheses; its value can be specified by the command option **-p** or the configuration parameter **Threshold** (a value in the command line has preference).

`Weight(sf=xx,min=yy)` – means that the algorithm considers weighted basis trees with a tree weight reflecting the relative importance of that tree (otherwise, if all basis trees are equally important, each weight is equal to 1). The weighting mode is controlled by parameter **BasisTreeWeight** in the configuration file. The number `xx` is a scaling factor used for the weight calculation (it is specified by the command line option **-f** or the configuration parameter **ScaleFactor**). The number yy is the minimum weight of a basis tree for the algorithm to consider that tree (value yy is determined by the configuration parameter **MinWeight**). Since a weight is non-negative, `min=0` means that all basis trees should be used.

`Closs=xx` – indicates the cost of a gene loss that is used by the algorithm to compute the quality of a species tree. The cost of loss is specified by the command line option **-l** or configuration parameter **CostOfLoss** (a setting in the command line has a preference).

`Cdup=xx` – indicates the cost of a gene duplication that is used by the algorithm to compute the quality of a species tree. The cost of duplication is specified by the command line option **-d** or configuration parameter **CostOfDuplication** (a setting in the command line has a preference).

`ParaPen=xx` – indicates the penalty for paralogy that the algorithm uses when constructing the optimal set of basis trees. The penalty is specified by the command line option **-y** or configuration parameter **ParalogyPenalty** (a setting in the command line is preferred).

`RShift=xx` – indicates a constant shift value used in phase 2 of the algorithm to find optimal candidate species to insert in the supertree. The shift value is specified by the command line option **-z** or configuration parameter **RShift** (a setting in the command line has a preference).

`SpA=xx` – indicates the architecture used for the table of quality value for each triple species topology. The possible `xx` options are `0d, 1d, 2d, 3d`. The architecture `3d` is the fastest one, but also the most memory-consuming, which can force to use slower options `2d` or even `1d` for solution of a task with numerous species. The desired architecture is selected by the command line option **-x** or the configuration parameter **SpecArrayDim** (also see Section 8.1).

`MPI-yyy` – appears in the log file, when the program works in a multiprocessor mode (`yyy` is a number of processors used). The prerequisite to parallel running of the algorithm is the presence of MPI environment version 1.2 or later **[2]**, and proper command line to run super3GL in this environment (see Section 3); in such situation the program automatically switches to the multiprocessor mode. The program can be prevented from the parallel execution with use of **-nompi** option in the command line.

When working in the normal mode, subsequent lines of the log file report the characteristics of source data (e.g. number of species, number and metrics of source trees, occurrence of species, etc.) and objects being constructed at Phase 1 such as set *P* of clades, “good” edges and nodes in the source trees, basis sets, and, finally, basis trees. All lines are time-stamped.

By default, the algorithm at Phase 2 does not print much on the console, but it outputs to the log file all steps of the supertree construction by induction in the form similar to as described in Section 5.1. In addition, a number of configuration parameters (their names start from `Log...`) are provided to get more data in the log file if desired (see Section 6.1).

5.3 File of basis trees

The basis tree file is used (i.e. created at Phase 1 and/or read at Phase 2), when its name is specified by the command line option **-b** or configuration parameter **`BasisName`**. The file name may include a path.

The basis tree file has the text format. Each line contains one binary rooted basis tree in parenthesis notation Newick **[3, 4]**. Leaves of a basis tree are species non-recurring in that tree. The trees are arranged within the file in the number of species ascending order.

The root of each tree has a numeric label and optional length. The meaning of the label is a cost of the basis tree, and the length equals a weight of the tree.

Examples of the basis tree files are available at the program web page **[5]**.

6. Program management

The primary method to control the `super3GL` program behavior is to set values of its parameters in the configuration file (Section 6.1), which allow the user to run the program without any parameters in the command line. Nevertheless, in some situations it can be desirable to have several different configurations or temporarily modify a parameter not changing the main configuration. In such cases the user should specify options in the command line (Section 6.2).

6.1 Configuration file

By default, the program looks for the configuration file named `super3GL.ini` in the current working directory (often but not always it is a directory, where the program executable resides and is invoked). If desired, another name and/or location of the configuration file can be specified by the command line option **`-c`**, for example:

```
super3GL -c ../example/myconfig.cfg
```

(here, the configuration file named `myconfig.cfg` resides in a directory `example` at the level of working directory). If the program cannot find a configuration file, it sets all parameters to their default values and outputs a warning message.
Note: When specifying paths to directories, the operating system rules have to be observed such as using of slash/backslash characters, enclosing a name in double quotes if it contains blanks or other special characters (when allowed), etc.

The configuration file is a usual text file containing lines of variable (virtually infinite) length. Space and tab characters within a line are ignored. Blank lines and lines that have any of the characters `# ; / *` as the first non-blank character are skipped in whole. A comment may also occur in a substantial line after the double slash; the program ignores it too.

Substantial lines of the configuration file have the following generic format:

```
key = value [,value2...]
```

which must be written in one line (continuation lines are not allowed). Values in the right side can be integer or real numbers in fixed and scientific notation (using `E` or `e` as a prefix for decimal exponent), character strings (if a string contains blanks or special characters, it must be enclosed in double quotes), or boolean values (true value can be specified as `1, true, yes, enable, on`; false value – as `0, false, no, not, disable, off`). Current version of the program does not use/permit more than one value per line yet.

The current version of `super3GL` program accepts only the keys listed below. When drawing up a configuration from scratch, it is suggested to specify keys in this order unless a group of connected keys is repeated. Default value is underlined if available.

`MaxValues = 3`

Maximum allowed number of values per key (not used in current version).

`WorkingDirectory =` *string*

Specifies a working directory, where the program looks for input files and writes output files. A path to the directory must be terminated with a slash or backslash depending on the operating system. If the string contains blanks or special characters, it must be enclosed in double quotes. This parameter is functionally equivalent to the command line option **`-w`** (with a preference of the command line setting).

`Milestones = 6`

This parameter controls the frequency of short message output to console that testify to the program operation. The value 0 is used to cancel the messaging; any other value is treated as the length of LSB mask of 1's for appropriate loop counter in the program (e.g. a number of trees read, basis sets built, etc.). When all bits of the counter selected by the mask become zeros, a message is output. For example, default value 6 means that messages will be displayed when the counter reaches 0, 64, 128, 192... The console output can slightly degrade the program performance so we suggest specifying value 0, when running the program at a cluster in batch mode. This parameter is functionally equivalent to the command line option **`-m`** (with a preference of the command line setting).

`Resume = true`

This parameter sets the program operation mode: "false" value means normal mode (Phase 1), and "true" value means resume mode (Phase 2). Note that a boolean value may be specified in many ways (see above). The operation mode can also be chosen with the command line option **`-r`**, which has a preference.

`MaxSpecies = 32`

Specifies the maximum number of species in the set of source gene trees. The parameter may be omitted only if a species table file is provided or the default maximum is sufficient. The value is quantized in steps of 32; if a value is not a multiple of 32, the program uses next greater multiple of 32.

`CSVdelimiter =` *character*

Specifies a character used as field delimiter in the species table (Section 4.2). Two special values can be used for nonprintable characters: `Space` for blanks and `Tab` for tabs. Default value is `Tab`.

`SkipLines = 0`

Specifies the number of initial lines in the species table for the program to skip.

`SpeciesTableName =` *string*

Specifies a name of the species table file. The name may include a path. Multiple species tables can be used if needed; in such case a group of the keys **`CSVdelimiter`**, **`SkipLines`**, **`SpeciesTableName`** occurs several times in the configuration file (and the program uses the aggregation of species from all tables). In addition, there is a command line option **`-s`** for the same purpose: species tables given in the command line are joined with those in the configuration file. To ignore all `SpeciesTableName` directions in the configuration file, one can use a command line option `--s`.

`MaxTrees = 16`

This optional parameter allows the user to inform the program about total number of gene trees in all input tree files. If omitted, the program increase allocated arrays as necessary, but this parameter can help in faster computation and more effective memory management.

`SpecLabelParts = 1`

Specifies the number of initial segments corresponding to a species name in the label of each leaf of input trees (see Section 4.1 for more detail).

`StopLabel` = *string* This parameter specifies a character to indicate the end of a leaf label (remainder of the label starting from that character is ignored). The *string* may contain more than one character e.g. `"@*"`, if input trees use different stop characters. By default, if the parameter is omitted, no stop character is used, and the entire leaf label will be considered.

`StrictLabelControl = false`

This parameter selects the leaf label check mode for the set of input trees. True value enables strict control of labels, false value disables some checks (see Section 4.1 for more detail).

`GeneTreeName` = *string*

Specifies a name of the input tree file (Section 4.1) that may include a path to the file directory. This parameter can occur more than once in the configuration file, if the set of input trees is represented by several files.

`GeneTree` = *string*

This parameter allows the user to insert a gene tree directly in the configuration file rather than in a separate file being referred to in the configuration. The tree has to be presented in the parenthesis notation Newick and written in one line. It is suggested to enclose whole Newick string in double quotes. This parameter can occur more than once in the configuration file, if multiple source trees need to be specified in this way.

`TreeWriteMode = 0`

This parameter controls details of the supertree presentation in the output file. By default, the supertree in parenthesis format contains only labels of the tree leaves. The parameter value can be a sum of the following options:

1:	output numeric labels for inner nodes of the tree;
2:	output character string labels for inner nodes of the tree;
4:	output the length of incoming edge for all inner nodes (including the root);
8:	output the length of incoming edge for all leaves;
16:	output hidden numeric labels for the tree leaves;
32:	output a numeric label and edge length only for the tree root.

For example, if value 5 is specified, then for each inner node and the root of the tree its numeric label and the length of incoming edge appear in the parenthesis notation. Recall that in `super3GL` the edge length means a validity of the daughter node.

`Super3Name` = *string*

Specifies a name of the file to write the supertree (Section 5.1). The name may include a path. If the file exists, it will be overwritten or updated depending on the program operation mode.

`Super3Sequence = true`

This parameter controls the output of results to the supertree file. If true value is specified, the supertree file contains the result of each induction step (Section 5.1); otherwise, the file contains only the final result, i.e. the supertree with all species.

`Super3Quality = false`

If a boolean value "true" is specified, before each (or single) tree line in the supertree file an additional comment string appears, which is described in Section 5.1.

`TreeLogMode = 0`

This parameter controls the format of any tree output in the log file (Section 5.2). By default, only labels of the tree leaves are included in the parenthesis notation. The value has the same meaning as in the parameter **`TreeWriteMode`**.

`LogActualTrees = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the whole set of input trees in parenthesis notation (after the tree pruning, if applied).

`LogActualSpecies = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the set of species that remain after the tree pruning.

`LogEntirePset = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the collection of sets of species, which are elements of the set *P*.

`LogBasisSets = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the collection of basis sets, which are elements of the set *P*.

`LogBasisTrees = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the sequence of basis trees built (one tree in Newick notation per line of the log file). The output form is the same as in the basis tree file (Section 5.3).

`LogTopology3 = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains the table of quality values for triple species topologies (only nonzero elements of the whole matrix).

`LogChoice3 = false`

If a boolean value "true" is specified, the log file (Section 5.2) contains detailed information on the choice of next species to insert in the current supertree during Phase 2 of the algorithm. For this information to be correctly understood, all trees in the log file should display the labels of inner nodes (see parameter **`TreeLogMode`**).

`LogSuper3 = true`

If a boolean value "true" is specified, the log file (Section 5.2) contains the entire sequence of the supertree building by induction. Otherwise, only the final supertree appears in the log file.

`BasisName` = *string*

Specifies a name of the basis tree file (Section 5.3); such name may include a path. The same information can be provided by option **`-b`** in the command line (which has a preference to the configuration setting).

`TreePruningMode = false`

This parameter controls a pruning of the source gene trees. True value enables the pruning, and vice versa. However, it is not a final choice as it depends on the **`Threshold`** configuration setting, which can also be specified in **`-p`** command line option. Despite this parameter, the algorithm removes all gene trees from the source set that involve only one species, because those trees do not bear any useful information.

`Threshold = 1`

This parameter controls a pruning of the source gene trees. It specifies a threshold value for the occurrence of a species. The value is considered only if the configuration parameter `TreePruningMode` is set to true). If a species occurs fewer times than the threshold value, this species is removed from all gene trees. Default value 1 says to remove only species not involved in the source tree (they could appear from the table of species). No tree pruning is carried out for the threshold of zero. The threshold value can also be specified in the command line option **`-p`** (which has a preference to the configuration setting).

`CostOfLoss = 2`

This parameter specifies the cost of a gene loss that is used by the algorithm to compute the quality of a species tree. This cost can also be specified by the command line option **`-l`** (which has a preference to the configuration setting).

`CostOfDuplication = 3`

This parameter specifies the cost of a gene duplication that is used by the algorithm to compute the quality of a species tree. This cost can also be specified by the command line option **`-d`** (which has a preference to the configuration setting).

`ExtendPset = false`

This parameter controls a composition of the set *P*. By default, the set contains all clades of all source trees and every combination of all but one species. If a true value of the parameter is specified, set *P* also contains many difference sets made of the default composition. The parameter can also be specified by the command line option **`-a`** (which has a preference to the configuration setting).

`BasisTreeWeight = true`

If a true value is specified, each basis tree is considers with a weight calculated during the construction of that tree. The weight can be from 0 to the **`ScaleFactor`** setting plus one. If a false value is specified, all weights equal 1.

`ScaleFactor = 10`

This parameter sets the value of a scaling factor, which is used for the calculation of a basis tree weight. The scaling factor can also be set in the command line option **`-f`** (which has a preference to the configuration setting).

`MinWeight = 0`

This parameter specifies a threshold for the basis tree weight. If a basis tree has the weight less than the threshold, such tree is not taken into account during the supertree construction in Phase 2 of the algorithm. Default value 0 means that all basis trees are considered independently of their weights.

`Uncertainty = -1`

This parameter allows the user to set a special value of the supertree node validity which is assigned to this node if the direct calculation leads to *x*/0 indeterminacy. In such situation, the specified value appears as the length of incoming edge for the node in parenthesis notation. The value can also be set in the command line option **`-u`** (which has a preference to the configuration setting).

`ZeroValue = 0.1`

This parameter allows the user to set a special value of the supertree node validity which is assigned to this node if the direct calculation gives 0. In such situation, the specified value appears as the length of incoming edge for the node in parenthesis notation. The value can also be set in the command line option **`-v`** (which has a preference to the configuration setting).

`EachStepReliability = false`

If a true value is specified, the algorithm in Phase 2 calculates the reliability of a species insertion in the current supertree at the last step of induction. Otherwise, the reliability is calculated only at the final step which can save some time. Command line options **`-e`** `/` **`--e`** do the same and have preference to this configuration parameter.

`SpecArrayDim = 3`

This parameter allows the user to select appropriate architecture for working data by making a choice in the performance/memory trade-off. A command line option **`-x`** has the same function. Default setting ensures the maximum performance, but it requires to store in memory a cubic matrix of float numbers with the dimension of a number of species. It is not always possible, and the user can be forced to reduce the memory demands by lowering of this parameter. Permitted values are 3, 2, 1, 0; recommendations for the choice are given in Section 8.

`ParalogyPenalty = 1.0` This parameter allows the user to set a penalty for paralogy that is used when constructing the set of basis trees. The value can also be set in the command line option **`-y`** (which has a preference to the configuration setting).

`RShift = 0.5` This parameter allows the user to set a constant shift of reliability that is used when inserting new species in the supertree on phase 2 of the algorithm. The value can also be set in the command line option **`-z`** (which has a preference to the configuration setting).

`SpeciesLength = true` If a true value is specified, the reliability of each vertex will be present in the resulting supertree as a length of the incoming edge to that vertex.

`SkipAmbiguos = true` If a false value is specified, each step of the algorithm at phase 2 will insert a species in the supertree so that the criterion $Q*(R + RShift)$ is maximized, where Q is the quality of a produced tree, R is the reliability of inserting that species, and *RShift* is specified by the parameter of the same name. Conversely, if a true value is specified, the species with $R = 0$ will be skipped despite the criterion; thus, some species may absent in the final supertree.

Examples of the configuration files with all possible parameters (some of them are commented-out) are available within the example tasks at the program web page **[5]**.

6.2 Command line options

Command line options are typically used to run `super3GL` with modified settings which were previously made in the configuration file or by default (the command line has a preference to the configuration). The options can have one of the following generic formats:

```
-key
--key
-key value
--key value
```

The value can be an integer or real number in the fixed or scientific notation (using `E` or `e` as a prefix for decimal exponent), a character or a character string. If a string contains blanks or special characters of the operating system, such string. The key and the value must be separated by at least one space. No space is permitted between the key and the dash(es). To get short help on command line options type **`-help`** or **`-h`** as a command line option.

Unless otherwise specified, the command line options may appear in any order. All keys are case-insensitive, character strings with a file name or location – depending on the operating system. Current version of the program accepts the following options in the command line (listed in the alphabet order; aliases are shown in brackets, if any):

`-a` `[-add]`

This option is functionally equivalent to the configuration parameter **`ExtendPset`** – it controls a composition of the set *P*. If this option presents, then additional clades are inserted in the set *P* independently of the configuration setting. If the option is specified in the form `--a`, then additional clades are not inserted to *P* independently of parameter **`ExtendPset`** in the configuration file.

`-b` *string* `[-basis | -bastree | -btree]`

This option has the same meaning as the configuration parameter **`BasisName`**: it specifies a name of the basis tree file.

`-c` *string* `[-conf | -config]`

This option allows the user to specify a name and directory of the configuration file (Section 6.1) as appropriate. Due to the configuration importance, this option should precede all other options, except for **`-w`** option, which may be the first one.

`-d` *number* `[-cd | -cdup | -dup]`

This option has the same meaning as the configuration parameter **`CostOfDuplication`**: it specifies a cost of the gene duplication event.

`-e` `[-each | -eachstep]`

This option has the same meaning as the parameter **`EachStepReliability`** in the configuration file. If this option is specified, the reliability of species insertion in the supertree is calculated at each induction step. If the option is specified in the form `--e`, the reliability is calculated only at final step.

`-f` *number* `[-sf | -factor | -scalefactor]`

This option has the same meaning as the configuration parameter **`ScaleFactor`**: it sets a scaling factor for calculation of the basis tree weight.

`-g` *строка* `[-log]`

This option allows the user to specify a name and directory of the log file (Section 5.2). By default, it is `super3GL.log` in the current working directory. If the option is specified in the form `--g`, the log file is not created.

`-h` `[-help]`
If a command line contains this option, the program ignores other options and outputs short help information.

`-l` *number* `[-cl | -closs | -loss]`
This option has the same meaning as the configuration parameter **`CostOfLoss`** : it specifies a cost of the gene loss event.

`-m` *number* `[-milestones | -stamp]`
This option has the same meaning as the configuration parameter **`Milestones`**: it controls the frequency of short messages to console testifying the program operation. If zero number is specified, no messages are displayed. Negative numbers are similar to their absolute values, but the short messages are displayed in all parallel branches in MPI mode (normally only to main console in the root branch).

`-nompi` `[--mpi]`
This option allows the user to prohibit running `super3GL` in parallel mode. The program will operate in uniprocessor mode. It can be helpful if an MPI environment is detected by mistake or does not function.

`-o` *string* `[-output | -supertree | -super3]`
This option has the same meaning as the configuration parameter **`Super3Name`**: it specifies a name and location of the supertree file (Section 5.1).

`-p` *number* `[-threshold | -minocc]`
This option has the same meaning as the configuration parameter **`Threshold`**: it specifies a threshold value for the occurrence of a species. If a species occurs fewer times than the threshold value, this species is removed from all gene trees.

`-r` `[-resume | -br | -basisread]`
This option has the same meaning as the configuration parameter **`Resume`** : it allows the user to select the program operation mode. If the option occurs in the command line, `super3GL` works in the resume mode irrespective of a mode chosen in the configuration file. If the option is specified in the form `--r`, the program works in normal mode also irrespective of the configuration setting. Otherwise, a configuration setting or a default mode is used.

`-s` *string* `[-spec | -species]`
This option has the same meaning as the configuration parameter **`SpeciesTableName`**: it specifies a name and location of the species table file (Section 4.2). The option may appear more than once in a command line, if the table of species consists of multiple files. <u>However, this option(s) must precede the option(s) which specifies the input tree file(s), if any.</u> Note that you cannot specify the delimiter character for a species table through the command line, so the default separator Tab is assumed. If a species table is also specified in the configuration file, the program uses a union of species tables from both sources. The user can prohibit using species tables from the configuration file by specifying this option in the form `--s`.

`-t` *string* `[-tree | -gt | -genetree]`

This option has the same meaning as the configuration parameter **`GeneTreeName`**: it specifies a name and location of the input tree file (Section 4.1). If it is necessary to join several tree files at input, the option may appear more than once in a command line. The program also joins trees specified through the configuration file unless this option is in the form `--t`., which ignores the trees given in the configuration.

`-u` *number* `[-uncertain | -uncertainty]`

This option has the same meaning as the configuration parameter **`Uncertainty`**: it sets a special value of the supertree node validity for situations where it cannot be calculated directly because of *x*/0 indeterminacy.

`-v` *число* `[-zerovalue | -value0]`

This option has the same meaning as the configuration parameter **`ZeroValue`**: it sets a special value of the supertree node validity for situations where its direct calculation gives zero value.

`-w` *string* `[-wd | -dir | -wdir]`

This option has the same meaning as the configuration parameter **`WorkingDirectory`** : it specifies a working directory, where the program looks for input files and writes output files. If used, the option should be the very first one in a command line, because it applies to all files. To prevent errors, we recommend not specifying a path together with a file name, when this option is used. A string in this option must be (back)slash-terminated.

`-x` *number* `[-dim]`

This option has the same meaning as the configuration parameter **`SpecArrayDim`**: it controls a structure the program uses to store working data. Only numbers 0, 1, 2, 3 are allowed; the bigger number, the faster and more memory-consuming program. See also Section 8.

`-y` *number* `[-para]`

This option has the same meaning as the configuration parameter **`ParalogyPenalty`**: it specifies the penalty value for paralogy in basis trees.

`-z` *number* `[-shift]`

This option has the same meaning as the configuration parameter **`RShift`**: it specifies a shift of the calculated reliability for insertion of a new species in the supertree.

7. Examples of the program usage

This section describes several examples of usage of the program `super3GL` for real tasks of different complexity. Source data and results for these examples are available at the program web page **[5]**. We do not touch a biological origin and ground of the tasks, and limit ourselves to formal aspects of the program use.

Four computing installations were used for solving of the tasks. We symbolize those computers as follows:

D1 A desktop workstation on the basis of single-core CPU Intel Pentium-IV with the frequency of 3 GHz and memory 2 GB. The program `super3GL` was running in uniprocessor mode (by specifying `-nompi` command line option).

D2	A desktop workstation on the basis of dual-core CPU Intel Core 2 Duo with the frequency of 2.5 GHz and memory 4 GB. The program was running in two-processor mode.
S32	A server on the basis of four eight-core CPUs Intel Xeon with the frequency of 2 GHz and memory 256 GB. The program (64-bit version) was running in up to 32-processor mode.
MVS	A cluster MVS-100K in the Joint Supercomputer Center of the Russian Academy of Sciences **[6]**. Computations were carried out with use of various numbers of processors up to 1024; each CPU had 2 GB memory.

7.1 Example 1 - 40 species

The source set of trees (file `input_trees.tre`) consists of 1000 binary trees of genes from 40 species. General characteristics of those trees are given in Table 7.1. The example does not use a species table, and use the configuration file `super3GL.ini`. Running times of the two algorithm phases for different computers are shown in Table 7.2. Specifically to the computer D2, the set of basis trees after the Phase 1 is given in the file `basis.tre`, the result – in the file `super3.tre`, the log – in the file `super3.log`. All these files are gathered into `example040.zip` archive downloadable from the program web page **[5]**.

7.2 Example 2 - 108 species

The source set of trees (file `new_trees.tre`) consists of 11516 binary trees of genes from 276 species, but those trees are pruned according to the species occurrence threshold of 500, which retains only 11184 trees and 108 species. General characteristics of those trees are given in Table 7.1. The example does not use a species table, and use the configuration file `super3GL.ini`. Running times of the two algorithm phases for different computers are shown in Table 7.2. Specifically to the computer MVS (with 32 processors involved), the set of basis trees after the Phase 1 is given in the file `basis.tre`, the result – in the file `super3.tre`, the log – in the file `super3.log`. All these files are gathered into `example108.zip` archive downloadable from the program web page **[5]**.

7.3 Example 3 - 276 species

The source set of trees is the same as in Example 2 (file `new_trees.tre` comprising 11516 binary trees of genes from 276 species), but tree pruning is not used in this case. General characteristics of those trees are given in Table 7.1. The example does not use a species table, and use the configuration file `super3GL.ini`. Running times of the two algorithm phases for different computers are shown in Table 7.2. Specifically to the computer MVS (with 128 processors involved), the set of basis trees after the Phase 1 is given in the file `basis.tre`, the result – in the file `super3.tre`, the log – in the file `super3.log`. All these files are gathered into `example276.zip` archive downloadable from the program web page **[5]**.

7.4 Example 4 - 814 species

This example uses a species table (Section 4.2) (the file `BacNames.csv`) containing 820 species. The source set of trees (file `all_trees.tre`) consists of 1511 binary gene trees. 6 species from the table do not occur in those trees, and two trees contain genes from only one species. After pruning with species occurrence threshold of 1, 1509 trees and 814 species retain. General characteristics of those trees are given in Table 7.1. The configuration file `super3GL.ini` is used. Running times of the two algorithm phases for different computers are shown in Table 7.2.

Table 7.1. Characteristics of source and intermediate data in Examples 1-4.

Parameter	Example 1	Example 2	Example 3	Example 4
Number of species	40	108	276	814
Number of genes	50932	146545	213370	38022
Number of gene trees	1000	11184	11516	1509
Number of genes per tree:				
minimum	31	2	7	15
maximum	60	124	164	123
average	51	13	19	25
standard deviation	6	11	16	14
Number of species per tree:				
minimum	30	2	3	5
maximum	40	48	88	122
average	31	7	11	24
standard deviation	2	5	9	13
Number of occurrences of a species in the source set of trees:				
minimum	980	500	25	1
maximum	1461	3140	3140	235
average	1268	857	527	46
standard deviation	126	495	417	38
Power of the set *P*	9750	37770	72290	15442
Total number of “good” edges	73552128	217998302	382098675	8293231
Number of good edges per element of *P*:				
minimum	980	396	25	1
maximum	22266	32122	43099	5013
average	7544	5772	5286	537
standard deviation	4095	3891	4510	588
Number of good edges per tree (including repeats):				
minimum	47819	1476	2949	1243
maximum	119308	260467	407419	30668
average	73552	19492	33180	5496
standard deviation	10624	17397	28200	3066
Number of basis sets	9190	36334	69046	14215
Total number of basis set splitting options	55287	112420	164286	21478
Maximum number of splitting options for a basis set	135	110	74	25
Number of basis trees	8784	33239	61995	11397
Maximum number of species in a basis tree	40	43	88	122
Basis tree weight:				
minimum	0.00	0.00	0.00	0.00
maximum	10.51	2.59	2.02	1.22
average	1.12	0.37	0.53	0.72
standard deviation	2.19	0.48	0.47	0.42

Table 7.2. Example 1-4 execution time on different computers (in minutes).

System	Number of CPUs	Phase	Example 1 (40 species)	Example 2 (108 species)	Example 3 (276 species)	Example 4 (814 species)
		Phase 1	23	243	646	99
D1	1	Phase 2	<1	3	263	–
		Total	23	246	909	–
		Phase 1	**11**	90	229	9
D2	2	Phase 2	**<1**	1	159	≈ 44000
		Total	**11**	91	388	30 days
		Phase 1	6	48	145	7
S32	4	Phase 2	<1	<1	79	22805
		Total	6	48	224	22812
		Phase 1	3	29	95	
S32	8	Phase 2	<1	1	37	–
		Total	3	30	132	
		Phase 1	3	17	42	2
S32	16	Phase 2	<1	<1	13	4992
		Total	3	17	55	4994
		Phase 1		**12**	29	1
S32	32	Phase 2	–	**<1**	9	2654
		Total		**12**	38	2655
		Phase 1		9	21	1
MVS	64	Phase 2	–	<1	6	1721
		Total		9	27	1722
		Phase 1		7	**14**	1
MVS	128	Phase 2	–	<1	**3**	1139
		Total		7	**17**	1140
		Phase 1		8	13	2
MVS	256	Phase 2	–	<1	4	790
		Total		8	17	792
		Phase 1			22	**1**
MVS	512	Phase 2	–	–	4	**390**
		Total			26	**391**
		Phase 1				17
MVS	1024	Phase 2	–	–	–	630
		Total				647
Note: Detailed results for the shaded cells are available at the program web page [5].						

Specifically to the computer MVS (with 512 processors involved), the set of basis trees after the Phase 1 is given in the file `basis.tre`, the result – in the file `super3.tre`, the log – in the file `super3.log`. Note that in this case the algorithm was unable to insert 82 species in the supertree due to ambiguous place of insertion (zero reliability). Then built supertree (with short names of species) is converted to the more friendly tree `super3n.tre` (with full names of Bacteria) using the command

```
        uncode super3.tre BacNames.csv super3n.tre.
```

All these files are gathered into `example814.zip` archive downloadable from the program web page [5].

8. Guidelines for use of super3GL

For the program to be used effectively, a proper setting of parameters is essential and should take dimensions and specifics of the task into account. The most of parameters described in Section 6 are unambiguously determined by source data and the user's needs. An exception is the configuration parameter **`SpecArrayDim`** (or, what is the same, command line option **`-x`**), which is discussed in Section 8.1. Another important issue is a choice of the number of processors for parallel execution of the program; this subject is considered in Section 8.2.

The `super3GL` usage experience suggests general recommendations to avoid common errors.

- The working directory is recommended to contain the configuration file (Section 6.1), input tree file(s) (Section 4.1) and the species table(s) (Section 4.2). Output files should be written to the same directory. The program executable, however, may reside in another directory: a path to it can be specified in the command line or inserted in the system list of automatically searched directories. The configuration parameter **`WorkingDirectory`** and command line option **`-w`** are intended for skilled users.

- It is suggested to make all parameter settings and mode selections in the configuration file. The command line is intended primarily to modify few settings made in the configuration.

- The user should not rely upon default values of the parameters because they can be changed in future versions of the program. More reliable way would be specifying in the configuration file all parameters and modes even if they are equal to the default settings.

- It is strongly recommended to use a species table (Section 4.2) containing only the species, which occur in the set of input trees, because it helps to save memory.

8.1 Selection of the data architecture

Depending on the memory available, the number of processor cores and the number of species in a task, the user can manage a speed–memory trade-off using the configuration parameter **`SpecArrayDim`** (or, what is the same, command line option **`-x`**). The maximum performance of the algorithm at Phase 2 is reached with the setting of `SpecArrayDim=3`, but not every task of practical interest can be solved with this setting.

For the parameter set to 3, Table 8.1 shows approximate limit of the number of species in various situations. If the number is insufficient for a given task, the configuration parameter can be specified as `SpecArrayDim=2`, but it may slow down the program performance at Phase 2 by 30-50%. The setting of 2 is sufficient for virtually any task, but if memory shortage is still detected, value of 1 can be used (performance of the program goes at least 3-4 times down). Note that a physical memory size is shown in the table and it does not consider a memory occupied by the operating system and other applications working at the same time. The exact memory amount also depends on the number and size of input trees so actual limits can be even less.

If a cluster consists of multi-core nodes equipped with rather small memory amount, so the memory is a bottle-neck, the user should take into account that Phase 2 of the program is highly scalable. This is why the program running in parallel on a greater number of cores can be more effective than using less cores per node aiming to have sufficient process memory to use `SpecArrayDim=3` configuration setting for a given task.

Table 8.1. Maximum number of species for the configuration setting `SpecArrayDim=3`.

Physical memory (MB)	Number of CPU cores used by the program							
	1	2	3	4	5	6	7	8
256	360	280	–	–	–	–	–	–
512	460	360	–	–	–	–	–	–
768	520	410	–	–	–	–	–	–
1024	580	460	400	360	–	–	–	–
1536	660	520	460	410	–	–	–	–
2048	730	580	500	460	420	400	380	360
3072	**830**	660	580	520	490	460	430	410
4096	**920**	730	630	580	530	500	480	460
6144	**1050**	**830**	730	660	610	580	550	520
8192	**1160**	**920**	**800**	730	670	630	600	580

Note: In order to work in the modes corresponding to shaded cells of the table, 64-bit operating system and processors are required. The cells with numbers shown in bold also require using of 64-bit version of `super3GL`.

8.2 Selection of the number of processors

One can make the following observations from the Table 7.2 data:

1) For a task with small number of species (less than 200), the main computational load is associated with Phase 1 of the algorithm. On the contrary, when the number of species exceeds 400, the complexity of Phase 2 quickly grows, and the user must resort to parallel processing.

2) The complexity of Phase 1 is mainly determined by total number of genes (i.e., leaves) in the set of input trees rather than by the number of species. On the contrary, the Phase 2 complexity depends on the number of species involved.

3) At Phase 1 of the algorithm, the parallelization effect ceases at a large number of processors; there is no sense in using more than 32-64 CPUs at the first phase.

4) Phase 2 of the algorithm can be parallelized effectively up to the number of processors approximately equal to the number of species, then the effect decreases.

Taking the above observations into account, a big task that involves many gene trees and many species is recommended to solve with *the phase separation*:

- First, run the program in normal mode at rather small number of processors such as 16 or 32 so that the file of basis trees is written (see Section 5.3), and cancel the operation when the file is ready. This stage of the work can be done on the user's workstation or a workgroup server provided that MPI environment are installed on that system. Running the program in uniprocessor mode is also possible, but it may take several hours. (Based on our experience, one hour is enough in most cases, if 16 CPUs are used).
- Then run the program in resume mode at a multiprocessor cluster. Several sessions may be required until the final supertree is built that includes all the species. If available, we recommend starting from the number of processors close to the number of species, or a smaller number. The user should be aware of the cluster node characteristics such as memory size and number of cores so that the computations could be planned taking Table 8.1 into

account. The time slot used for calculation must be sufficient for at least one step of induction, which can be estimated after few initial steps.

8.3 Troubleshooting

During the first phase, `super3GL` can display various messages about suspicious situations or errors. These messages are usually caused by errors in the source data and/or program parameters. Text of the message normally gives enough information to find and fix the error.

Error conditions at Phase 2 are not always indicated by clear messages, especially in a multiprocessing mode. Sometimes an error or exception occurs in the operating system or MPI environment itself. Their often reasons are mistiming of processes, short-term equipment failures, timeouts or faults in the communication infrastructure. The program bugs are also possible. In such situations, the following order of actions is suggested:

1) ensure that the last version of `super3GL` is used, which is available at the web page **[5]** (may be the bug is already fixed);
2) check if the error may be caused by running out of memory: determine the occupied memory size with use of operating system tools, and compare it with physical memory available. On the Windows platform, when the occupied memory size exceeds 2 GB per process, switch to 64-bit versions of the program and operating system;
3) rerun the program in resume mode without any modification of other parameters, and continue the supertree construction starting from the last unfinished step of induction;
4) continue the computation using different number of processors;
5) not touching the basis tree file (Section 5.3), delete the partially built supertree file (Section 5.1) and run the program again in resume mode at different number of processors, in order to start the supertree construction from the very beginning.

If these recommendations do not help, address to the software developer (Dr. Lev Rubanov, rubanov@iitp.ru) with indication of the program version number and type (32/64-bit), used command line, system message (if any), and attached input/output data files and the configuration file.

9. References

1. Lyubetsky V.A., Rubanov L.I., Rusin L.Yu., Gorbunov K.Yu. “Cubic time algorithms of amalgamating gene trees and building evolutionary scenarios”. *Biology Direct*, 2012, 7:48.

2. MPI: A message-passing interface standard [http://www.mpi-forum.org/docs/mpi21-report.pdf]

3. Newick format - Wikipedia [http://en.wikipedia.org/wiki/Newick_format]

4. Gary Olsen's Interpretation of the "Newick's 8:45" Tree Format Standard [http://evolution.genetics.washington.edu/phylip/newick_doc.html]

5. A program for supertree construction [http://lab6.iitp.ru/en/super3gl/]

6. Joint Supercomputer Center of the Russian Academy of Sciences [http://www.jscc.ru/]