IlpGeneratorGGL user's manual

(authors K.Yu. Gorbunov, V.A. Lyubetsky,
developer: R.A. Gershgorin)

2017

Overview

The phylogenetic tree with chromosome structures in the leaves is given. It is required to assign the structures to internal nodes of the tree to minimize the total distance between end structures of each edge. Unequal gene content and paralogs in all nodes are allowed. Paralogs should be identified such that the total distance for all resulting structures without paralogs is minimal.

The task can be reduced to Integer Linear Programming (ILP), the details can be found in [1].

Due to the large quantity of variables and constraints in the corresponding ILP problems, the IlpGenerator utility has been developed, it allows to produce description of ILP problem, suitable for external ILP solvers (such as IBM CPLEX).

Building of IlpGeneratorGGL from source code

The IlpGeneratorGGL is implemented using C++. The source code can be found on http://lab6.iitp.ru/en/ilp_generatorggl/. To build binary from source, several steps are required.

1. CMake file is provided alongside the source code. To generate the solution, cmake utility should be installed (installers for various platforms can be found at https://cmake.org/download/)
2. The source code depends on Boost libraries. The boost library should be install prior to project building. For Unix (was tested on Ubuntu 16.04) the command “sudo apt-get install boost” will do the trick. For Windows the Boost libs can be built from source code and used for building IlpGeneratorGGL. The libs, built for Windows, should be placed to the correct place relatively the CMakeLists.txt. The relative path to libs and includes can be changed in cmake variable ${EXTERNAL_LIBS}. This directory should contain boost directory with bin, lib and lib64 subdirectories, which should contain corresponding libs and includes.
3. To build the utility go to the directory with CMakeLists.txt file and run command: cmake -G “Unix Makefiles” (the -G parameter means the type of project to be generated, for example, to build visual studio 2013 project, the generator should be “Visual Studio 12 2013”, other generators can be obtained from the command “cmake –help”).

4. After project files have been generated, we need to build binary. If Unix generator is used, run the command “make”. The project should be built and binary should appear in the “bin” directory. For visual studio generator, it is required to open solution and build it in Visual Studio.

IlpGeneratorGGL binary usage

The binary does not need to be installed.

The binary has the following command line parameters:

- --input (-i) – input file with the description of a task (see next section for details).
- --output (-o) – output file with the description of the corresponding ILP task in IBM lp format.
- --mapping (-m) – output file with correspondence between gene names and their numeric codes.

See *Output files description* section for details on output and mapping options.

Input file format description

The first line contains the description of the tree in parentheses format. Names of leaves are provided before “:” sign, node numeric codes are provided after “:” sign.

Example of tree description:

((c:4,d:5):2,(e:6,f:7):3):1;

Next part of the file contains the values of $s(.)$ function, that is maximal number of paralogs for genes with certain indexes. The first line of the section contains the number N of genes. Next N lines contain the index of gene and $s(.)$ value, separated by space.

Example of $s(.)$ description:

2

1 3

2 1

Next part of the file contains the description of the gene structures. The format of this section is identical to the format, used in ChromoGGL program, the description can be found at ChromoGGL manual (http://lab6.iitp.ru/en/chromoggl/chromo_manual_en.pdf#page=17).

Example of structures section:

4;

c; 4; 1; C3: +1.1+2.1-1.2;

d; 5; 1; C3: +1.1+1.2-2.1;

e; 6; 1; C3: +2.1+1.1-1.2;

f; 7; 1; C3: +1.1+1.2+2.1;

Full version of input file example can be found here.

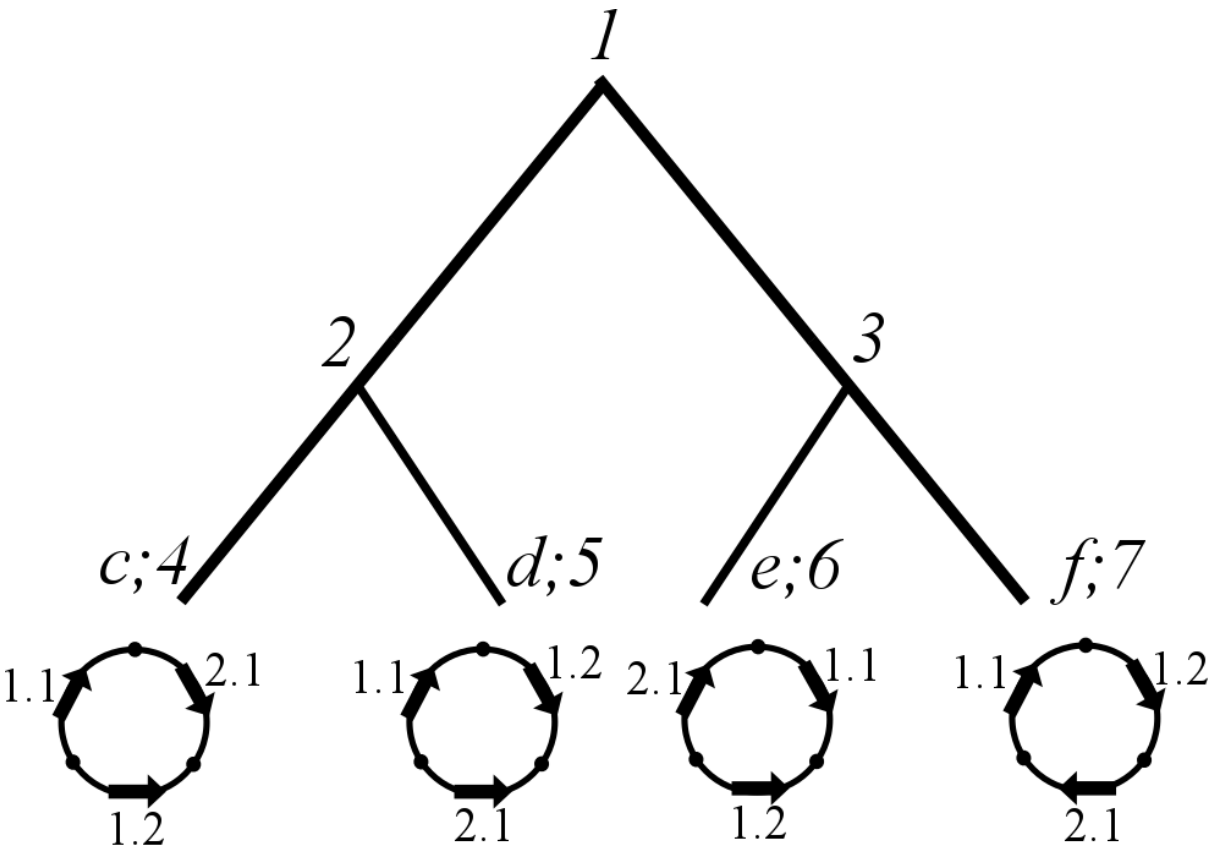

Figure 1. Initial chromosome structures from example

Output files description

The output of IlpGeneratorGGL consists of two files:

- File with ILP task in IBM lp format, the description of file format can be found at https://www.ibm.com/support/knowledgecenter/en/SS9UKU_12.4.0/com.ibm.cplex.zos.help/FileFormats/topics/LP.html

```
Minimize
        + x_b34  + x_a24  + x_a22  + x_a21  + 2 y_b24  + ...

Subject To
      z_111 + z_112 <= 1
      z_121 + z_122 <= 1
      z_111 + z_121 <= 1
      ...
      o_5 + z_111 + ... >= 1
      o_6 + z_412 + z_511 + z_512 + z_311 + z_321 >= 1

Bounds
      0 <= u_a13 <= 1
      0 <= u_a31 <= 2
      0 <= u_a12 <= 3
      ...

General
      u_a13
      u_a31
      u_a12
      ...

Binary
      z_111
      z_112
      z_121
      ...

End
```

Figure 2. Example of generated task file. Full version can be found *here*

- File with mapping of gene names to indexes. For sake of simplicity gene names are replaced by integer indexes, this file helps to restore names after solution is found (see next section for details on file usage). File contains several lines, the number of lines is equal to the number of different genes, each line contains gene name and corresponding index, separated by space symbol. See the example for two genes below:

  gene1 1

  gene2 2

Full pipeline description for structure reconstruction.

Alongside the IlpGeneratorGGL utility we provide the set of scripts for structure reconstruction:

- *solution_verifier.py* – python script, that verifies the solution, provided by ibm cplex optimizer. Can be used as command line utility:
  python solution_verifier.py -s result.sol -l task.lp
  here *result.sol* is output file from cplex, *task.lp* – file with task, generated by IlpGeneratorGGL. The result of the script is the set of constraints, that are not true for current solution. If solution is correct, "No errors found in solution" will be printed to console.
- *inner_structs_extractor.py* – python script, that extracts structures in inner nodes of the tree. Can be used as command line utility:
  python inner_structs_extractor.py -s result.sol -i map.txt -o structs.txt
  here *result.sol* is output file from cplex, map.txt – file, that provides mapping from internal indexes to real gene names (this file is generated by IlpGeneratorGGL), *structs.txt* – file with extracted structures.

Solution of the test example:

The solution for the initial task, processed by scripts (the set of structures, extracted by script can be found here), gives such a tree:

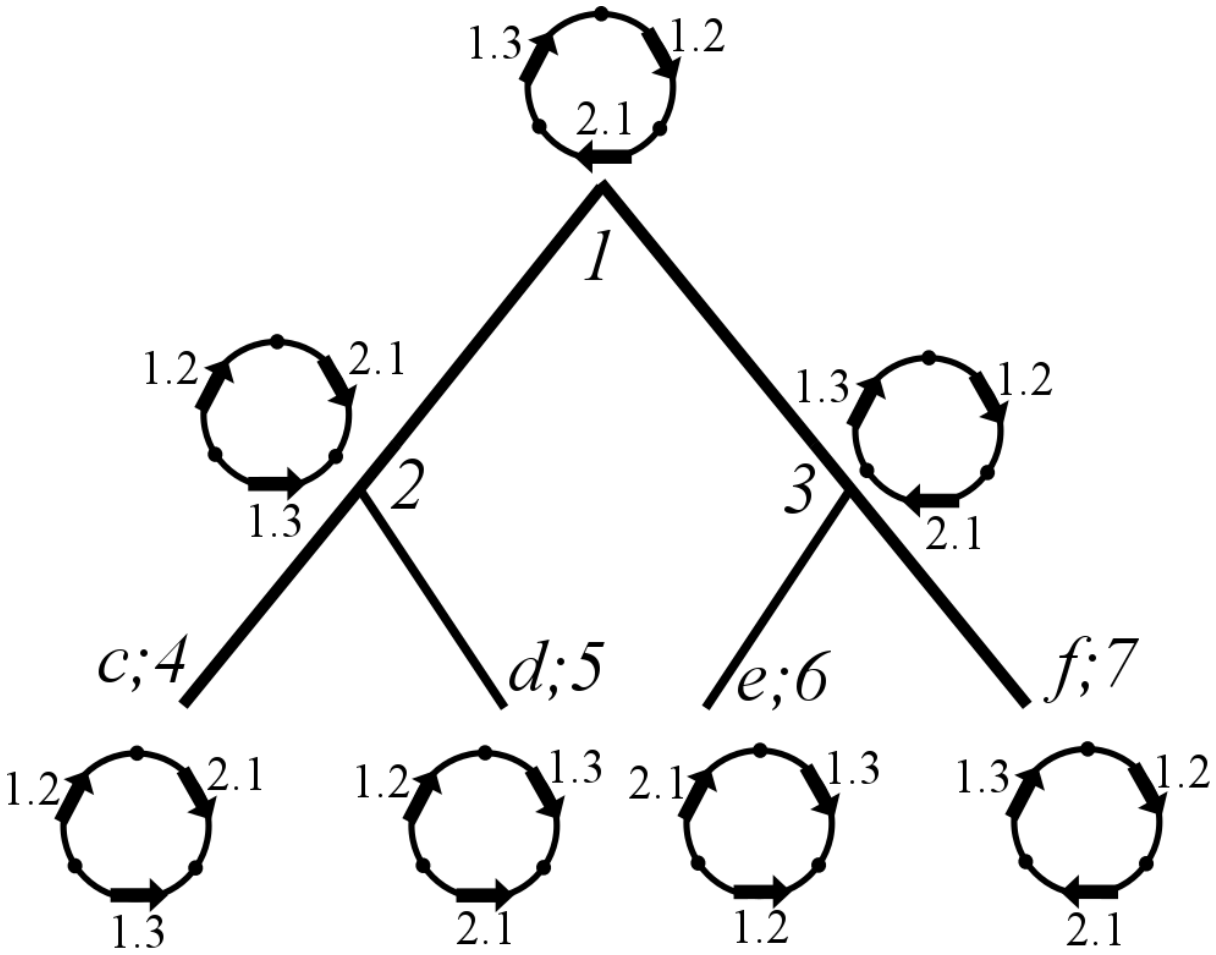

Figure 3. The solution for the test example

References

[1] V.A. Lyubetsky, R.A. Gershgorin, K.Yu. Gorbunov. Chromosome structures: reduction of certain problems with unequal gene content and gene paralogs to integer linear programming. BMC Bioinformatics, 2017