On Integer and Bilevel Formulations for the k-Vertex Cut Problem

Code Documentation

Fabio Furini[1], Ivana Ljubić[2], Enrico Malaguti[3] and Paolo Paronuzzi[3]

[1] *LAMSADE, Université Paris-Dauphine, 75775 Paris Cedex 16, France*
fabio.furini@dauphine.fr
[2] *ESSEC Business School of Paris, Cergy-Pontoise, France*
ivana.ljubic@essec.edu
[3] *DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy*
enrico.malaguti@unibo.it

Structure

The code has been implemented in the *C++* programming language and uses the *g++* compiler (version 5.4.0). We used `CPLEX 12.7.1` and the `Concert Technology` framework to implement our branch-and-cut algorithms.
The folder named *src* contains all the source and header files. The folder named *obj* is the folder where all the object files are generated. The input instance must be in *Dimacs* format. The following is an example of a simple instance:

```
p edge 3 3
e 1 2
e 1 3
e 2 3
```

The file named *Makefile* contains the instruction to compile the program. Inside the file there are two variables that the user could have to change: variable *SYSTEM* defines the operating system and variable *CPLEXDIR* defines the directory where Cplex is installed. Once these variables are properly defined, the user can build the program using the command *make*.
The executable file is named *KSEP* and it receives as input 9 parameters:

1. path of the instance;
2. used formulation to solve the problem;
3. option to solve the linear relaxation (set to 1 if you want to solve the linear relaxation of the problem);
4. value of k (number of subsets in which the graph must be divided);

5. absolute tolerance to consider a cut as violated;
6. time limit (in seconds);
7. frequency in calling the procedure to detect violated cuts in fractional solutions;
8. option to use the version with weights (set to 1 if you want to use weights);
9. name of the output file where all the relevant information is reported.

The possible values for parameter number 2 are:

- 1 for the formulation defined as $COMP$;
- 2 for the formulation defined as REP;
- 22 for the formulation defined as REP_{lp};
- 4 for the formulation defined as NAT;
- 44 for the formulation defined as NAT_s;
- 7 for the formulation defined as HYB.

An example of how to run the program is:

```
./KSEP instance_path 22 0 5 0.5 3600 100 0 Output.txt
```

The program prints a file of output with the following information:

- path of the instance;
- number of vertices of the graph;
- number of edges of the graph;
- the number corresponding to the used formulation;
- the name of the used formulation;
- value of k (number of subsets in which the graph must be divided);
- absolute tolerance to consider a cut as violated;
- frequency in calling the procedure to detect violated cuts in fractional solutions;
- number of vertices assigned to the k-vertex-cut by the preprocessing;
- best integer solution value;
- best bound value;
- computation time;

- status of *CPLEX*;
- number of variables;
- number of constraints;
- number of B&B nodes;
- number of user cuts;
- number of lazy cuts.

An example of a line of output is:

```
instance_path 34 78 22 REP_lp 5 0.5 100 0 2 2 0.04135 Optimal 68 147 97 6 177
```

When the linear relaxation is solved, in addition to the input information, only the solution value and the computation time are printed:

```
instance_path 34 78 22 REP_lp 5 0.5 100 0 0.554722 0.006079
```

Description of files

The *src* folder contains the following files:

- *StdPath_TerminalModel.cpp* and *StdPath_TerminalModel.h* contains the implementation of the formulation defined as REP;
- *StdPath_withLongPath.cpp* and *StdPath_withLongPath.h* contains the implementation of the formulation defined as REP_{lp};
- *BilevelModel.cpp* and *Bilevel.h* contains the implementation of the formulation defined as NAT;
- *BilevelModel_withLeaf.cpp* and *BilevelModel_withLeaf.h* contains the implementation of the formulation defined as NAT_s;
- *SmartModel.cpp* and *SmartModel.h* contains the implementation of the formulation defined as HYB;
- *global_functions.cpp* and *global_functions.h* contains the implementation of function used by more than one algorithms;
- *global_variables.cpp* and *global_variables.h* contains the definition of global variables;
- *Graph_v4.cpp* and *Graph_v4.h* contains the structures to support the graph;
- *LP_Model.cpp* and *LP_Model.h* contains the implementation of the linear relaxation of the defined formulation;
- *main.cpp* is the main file.