How to use the Round-robin to Graph Colouring Converter

Rhyd Lewis, October 2010

This folder contains the C++ source files for the round-robin to graph colouring generator described in:

- *Lewis, R. and J. Thompson (2011) 'On the Application of Graph Colouring Techniques in Round-Robin Sports Scheduling'. Computers and Operations Research, vol. 38(1), pp 190-204.*

The code is such that it can be compiled in both Linux and Windows. For example, in Linux you could use the following command:

```
g++ *.cpp –o rrgen
```

Once compiled, the program can be run at the command line. An example command to get started would be:

```
rrgen 20 out 1234
```

where `rrgen` is the name of the program; `20` is the size of the problem (20 teams); `out` is the name of the output file; and `1234` is a random seed.

The program will then ask the user a series of questions. “0” corresponds to “no”. Note that there is very little error checking of user input, so if the program does something strange, check what you have typed and start again. The choices are as follows:

- Should the round robin be single (0) or double (1)?
- output to screen (0/1)?
- add further constraints (0/1)?
    - *(if yes)*
    - perfect solution? (0/1)?
        - *(if yes)*
        - enter name of reference solution?
    - proportion of preassignments? [number between 0 and 1]
    - proportion of match/round unavailabilities? [number between 0 and 1]
- add comments to output file (0/1)

The number of vertices |V| in the generated files are as follows:

- Single round-robin, no additional constraints: (n(n-1))/2
- Double round-robin, no additional constraints: n(n-1)
- Single round-robin, additional constraints: (n(n-1))/2 + (n-1)
- Double round-robin, additional constraints: n(n-1) + 2(n-1)

Note that if we choose to add additional constraints and want to ensure the existence of a feasible round-robin, we also need to supply a reference solution to the program. A reference solution should specify a feasible round-robin solution in a text file and should be determined beforehand. Thus, when the program goes about adding constraints, it will do so in such a way that the reference solution obeys all of these. Two example files have been provided **16.dob.sol** and **30.dob.sol,**

specifying solutions for double round-robin problems with n=16 and 30 respectively. In these files the first line specifies the number of vertices |V| in the graph colouring solution. Then, for each vertex, 1,…,|V| an integer is given, specifying the colour of the vertex. Note that no match vertices should be included in the reference solution files: thus, they represent solutions to round-robins with no additional constraints.

Here is an example run, producing the output file **out.dob**, which defines a double round-robin graph with 16 teams, extra constraints, and a guaranteed feasible round-robin solution.

```
rrgen 16 out 1234

Should the round robin be single (0) or double (1)?: 1
Output to screen (0/1)?: 0
Add further constraints (0/1)?: 1
Perfect solution? (0/1)?: 1
Enter name of reference solution? 16.dob.sol
Proportion of preassignments?: 0.25
Proportion of match/round unavailabilities?: 0.1
Add comments to output file (0/1) 1
```