Solving Paint-By-Numbers Puzzles with GLPK

Andrew Makhorin <mao@gnu.org>

August 2011

1 Introduction[1]

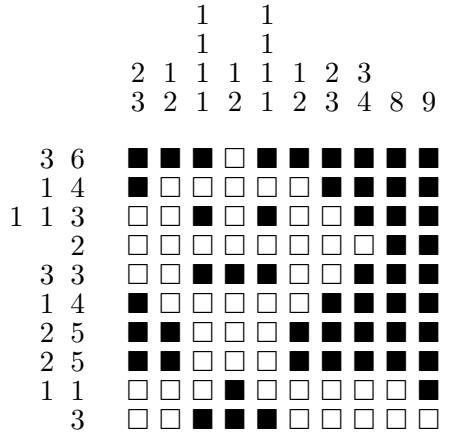
A *paint-by-numbers* puzzle consists of an $m \times n$ grid of pixels (the *canvas*) together with $m + n$ *cluster-size sequences*, one for each row and column. The goal is to paint the canvas with a picture that satisfies the following constraints:

1. Each pixel must be blank or white.

2. If a row or column has cluster-size sequence s_1, s_2, ..., s_k, then it must contain k clusters of black pixels—the first with s_1 black pixels, the second with s_2 black pixels, and so on.

It should be noted that "first" means "leftmost" for rows and "topmost" for columns, and that rows and columns need not begin or end with black pixels.

Example



[1]This section is based on the material from [1].

2 Solving a puzzle

The Paint-By-Numbers puzzle can be formulated as a 0-1 integer feasibility problem. The formulation used in GLPK was proposed in [1].

For solving puzzles there are used two components, which both are coded in the GNU MathProg modeling language [2]: the model section and the data section. The model section is common for all puzzles and placed in file `pbn.mod`. This file is included in the GLPK distribution and can be found in subdirectory `examples/pbn`.

To solve a particular puzzle the user only needs to prepare the data section, which defines input data to the puzzle. The data section for the example puzzle from the previous section may look like follows (here `m` is the number of rows, and `n` is the number of columns):

```
data;

param m := 10;

param n := 10;

param row : 1  2  3  :=
      1     3  6  .
      2     1  4  .
      3     1  1  3
      4     2  .  .
      5     3  3  .
      6     1  4  .
      7     2  5  .
      8     2  5  .
      9     1  1  .
      10    3  .  .
;

param col : 1  2  3  4  :=
      1     2  3  .  .
      2     1  2  .  .
      3     1  1  1  1
      4     1  2  .  .
      5     1  1  1  1
      6     1  2  .  .
      7     2  3  .  .
      8     3  4  .  .
      9     8  .  .  .
      10    9  .  .  .
;

end;
```

Let the data section for a puzzle be placed in file `foo.dat`. Then to solve the puzzle the user should enter the following command:

```
glpsol --minisat -m pbn.mod -d foo.dat
```

This command invokes `glpsol`, the GLPK LP/MIP stand-alone solver, which reads the model section from file `pbn.mod`, the data section from file `foo.dat`, translates them to an internal representation, and solves the resulting 0-1 integer feasibility problem. The option `--minisat` tells `glpsol` to translate the feasibility problem to a CNF satisfiability problem and then use the MiniSat solver [3] to solve it.

If a solution to the puzzle has been found, that is indicated by the message `SATISFIABLE`, `glpsol` prints the solution to the standard output (terminal), writes it to file `solution.ps` in the PostScript format, and also writes it to file `solution.dat` in the form of MathProg data section, which can be used later to check for multiple solutions, if necessary (for details see the next section). The message `UNSATISFIABLE` means that the puzzle has no solution.

Usually the time taken to solve a puzzle of moderate size (up to 50 rows and columns) varies from several seconds to several minutes. However, hard or large puzzles may require much more time.

Data sections for some example puzzles included in the GLPK distribution can be found in subdirectory `examples/pbn`.

3 Checking for multiple solutions

Sometimes the user may be interested to know if the puzzle has exactly one (unique) solution or it has multiple solutions. To check that the user should solve the puzzle as explained above in the previous section and then enter the following command:

```
glpsol --minisat -m pbn.mod -d foo.dat -d solution.dat
```

In this case `glpsol` reads an additional data section from file `solution.dat`, which contains the previously found solution, activates an additional constraint in the model section to forbid finding the solution specified in the additional data section, and attempts to find another solution. The message `UNSATISFIABLE` reported by `glpsol` will mean that the puzzle has a unique solution, while the message `SATISFIABLE` will mean that the puzzle has at least two different solutions.

4 Solution times

The table on the next page shows solution times on a sample set of the paint-by-numbers puzzles from the `<webpbn.com>` website. This sample set was used in the survey [4] to compare efficiency of existing PBN solvers.

The authors of some puzzles from the sample set have given permission for their puzzles to be freely redistributed as long as the original attribution and copyright statement are retained. In the table these puzzles are marked by an asterisk (*). The files containing the MathProg data sections for these puzzles are included in the GLPK distribution and can be found in subdirectory `examples/pbn`.

All runs were performed on Intel Pentium 4 (CPU 3GHz, 2GB of RAM). The C compiler used was GCC 3.4.4 with default optimization options.

The column 'Sol.Time' shows the time, in seconds, taken by the `glpsol` solver to find a solution to corresponding puzzle. The column 'Chk.Time' shows the time, in seconds, taken by `glpsol` to check for multiple solutions, i.e. either to prove that the puzzle has a unique solution or find another solution to the puzzle. Both these times do not include the time used to translate the MathProg model and data sections into an internal MIP representation, but include the time used to translate the 0-1 feasibility problem to a CNF satisfiability problem.

References

[1] Robert A. Bosch, "Painting by Numbers", 2000. `<http://www.oberlin.edu/~math/faculty/bosch/pbn-page.html>`.

[2] GLPK: Modeling Language GNU MathProg. Language Reference. (This document is included in the GLPK distribution and can be found in subdirectory `doc`.)

[3] Niklas Eén, Niklas Sörensson, "An Extensible SAT-solver", Chalmers University of Technology, Sweden. `<http://minisat.se/>`.

[4] Jan Wolter, "Survey of Paint-by-Number Puzzle Solvers". `<http://webpbn.com/survey/>`.

Table 1: Solution times on the sample set of puzzles from [4]

Puzzle		Size	Notes	Sol.Time, s	Chk.Time, s
#1	Dancer*	10×5	L	< 1	< 1
#6	Cat*	20×20	L	< 1	< 1
#21	Skid*	25×14	L, B	< 1	< 1
#27	Bucks*	23×27	B	< 1	< 1
#23	Edge*	11×10		< 1	< 1
#2413	Smoke	20×20		< 1	< 1
#16	Knot*	34×34	L	1	1
#529	Swing*	45×45	L	1	1
#65	Mum*	40×34		1	1
#7604	DiCap	55×55		10	10
#1694	Tragic	50×45		3	3
#1611	Merka	60×55	B	4	4
#436	Petro*	35×40		1	1
#4645	M&M	70×50	B	5	6
#3541	Signed	50×60		7	7
#803	Light*	45×50	B	1	1
#6574	Forever*	25×25		1	1
#2040	Hot	60×55		6	6
#6739	Karate	40×40	M	2	2
#8098	9-Dom*	19×19		1	2
#2556	Flag	45×65	M, B	2	2
#2712	Lion	47×47	M	11	12
#10088	Marley	63×52	M	135	226
#9892	Nature	40×50	M	850	1053

* Puzzle designer has given permission to redistribute the puzzle.

L Puzzle is line solvable. That is, it can be solved one line at a time.

B Puzzle contains blank rows or columns.

M Puzzle has multiple solutions.