TCG HW1 Report

R05921058
楊承翰

1. How to compile and run.
   To compile, use “make”, “make P”, “make I”, and “make PI” to compile four different executables with different algorithms. The basic algorithm (made by “make”) is simple DFS, adding “P” means doing preprocessing, and “I” means doing implication, which will be described in the next section.
   To run, use the command “./solver n < input”, where “solver” can be replaced by other executables, and n is the size of the puzzle.

2. Algorithm
   The basic DFS algorithm enumerates every undecided grid with black or white. After assigning a color, it checks if the current configuration contradicts to the constraints. If there is no violations, continue DFS until a solution is found, otherwise backtrace to the last decision. There are two optimizations that may speed up the program:
   a. Preprocessing
      Before starting DFS, we can observe the constraint and know that some grids must be black or white. This can help to reduce some decisions and speed up the process.
   b. Implication
      After assigning a grid to a color, the color of some other grids are also decided, and those grids may further imply other grids as well. The implication process assigns these grids to their implied colors and can also reduce search space.

3. Experiment result
   The testdata are generated by the command “./boardgen n 100 0.7 0.3 12345”, and timeout limit is set to 300 seconds. “P” means preprocess and “I” means implication.

	DFS	DFS + P	DFS + I	DFS + P + I
n = 10	0.51 s	0.51 s	0.03 s	0.03 s
n = 11	4.38 s	4.38 s	0.07 s	0.06 s
n = 12	13.4 s	13.5 s	0.09 s	0.09 s
n = 13	21.8 s	21.9 s	0.11 s	0.11 s

n = 14	timeout	timeout	0.76 s	0.76 s
n = 15	timeout	timeout	0.68 s	0.63 s
n = 16	timeout	timeout	15.6 s	14.6 s
n = 17	timeout	timeout	14.3 s	14.3 s
n = 18	timeout	timeout	140 s	143 s
n = 19	timeout	timeout	timeout	timeout

4. Discussion

   The game is proved to be NP-complete. We can find out that in most cases the solver can solve very fast, but there are a few cases that may cause a long run time. Also, puzzles with fewer 1's are generally harder than those with more 1's, because fewer implications take place.

   From the experiment result, we can see that preprocessing is not very effective, sometimes even makes the program slower because of the overhead. However, implication is very effective. With implication, we can solve 100 15*15 puzzle in less than a second.