Simulated Annealing - Report

Nicolas Boileau, Simon Stastny

October 8, 2012

1 Description of implementation

We decided to implement just one of the puzzles: egg carton puzzle. This report covers this only.

1.1 Representation

To represent our system, we use an array of booleans with 2 dimensions: boolean[x][y]. When there is an egg, the boolean is true. The temperature is an integer that starts at 2000 and decrease slowly at a rate of 95% per run until it reaches the minimal temperature of 1.

For each temperature, there are 20 iterations of finding solution and accepting/rejecting it. This number is `ITERATION_RUNS` parameter of `SimulatedAnnealing` class and can be modified (however we found 20 to be sufficient for this puzzle).

1.2 Objective function

To evaluate a solution, we first create a number that we consider as the maximum number of eggs that we could possibly have in the box if we respect the constraints. That doesn't mean that there is a solution in which we can put that many eggs, but we are sure that we can't put more. This number is

$$max = size\_of\_the\_box \times limit\_of\_eggs\_per\_line$$

For example for a box with a size of 6*6 and a constraint of K=2, we consider that we can never place more than 6*2 = 12 eggs.

Then, for a given solution, we count the `number of eggs in the box`, and we retrieve the `number of violations`, which is the number of eggs that are not respecting the contraints. For example if on a same line we have 4 eggs, with a constraint K=2, we count 2 violations.

Finally, we retrieve this number to the first number we calculated. So for a given solution X, we have a number calculated using the following formula:

$$energy(X) = max - (eggCount() - violationCount());$$

The smaller this number is, the closer we are from the optimal solution. If we reach 0, then we are sure that we have an optimal solution. If there is at least one violation, then the solution is not considered as valid. But we can still adopt it as one of our node in our algorithm.

Now that we have our objective function, let be X our current best solution and X' our new candidate. If X' is better (energy(X') < energy(X)), then we select it as our new best solution. But even if it's not the best, we have a probability to accept it. To decide whether we adopt a solution X' or not, we generate the probability of accepting it at a temperature T using the following formula:

$$P = \exp(\frac{1000 \times (energy(X) - energy(X'))}{T})$$

Then we generate a random double between 0 and 1, if it's less than the probability P we just calculated, then we accept the solution X' even though it's not better than our current solution.

1.3 Neighbour generation

To generate a neighbour from a current state, we check whether the solution is valid (no violation of the constraints) or not. If it is valid, we add one egg randomly in a free space. If it is not valid, then we move one random egg to a random free space. Thus, the difference between a state and one of its neighbour will always be one egg added or moved.

2 Solutions found

2.1 Egg Carton puzzle K=2, M=5

There can be 12 eggs placed in this variant of the puzzle.

```
X _ _ X _
_ X X _ _
X _ _ _ X
_ _ X _ X
_ X _ X _
```

2.2 Egg Carton puzzle K=2, M=6

There can be 12 eggs placed in this variant of the puzzle.

```
_ _ _ X X _
X X _ _ _ _
_ _ _ _ X X
X _ X _ _ _
_ X X _ _ _
_ _ _ X _ X
```

2.3 Egg Carton puzzle K=1, M=8

There can be 8 eggs placed in this variant of the puzzle.

```
_ _ _ _ X _ _ _
X _ _ _ _ _ _ _
_ _ _ _ _ _ _ X
_ _ _ _ _ X _ _
_ _ X _ _ _ _ _
_ _ _ _ _ _ X _
_ X _ _ _ _ _ _
_ _ _ X _ _ _ _
```

2.4 Egg Carton puzzle K=3, M=10

There can be 30 eggs placed in this variant of the puzzle.

```
_ X _ _ _ _ X _ X _
_ _ X _ X X _ _ _ _
X _ X _ _ _ _ X _ _
_ X _ _ _ _ _ X _ X
_ _ _ X _ _ _ _ X X
_ _ _ X X _ X _ _ _
X X _ _ _ _ _ _ _ X
_ _ X _ X _ _ X _ _
X _ _ X _ X _ _ _ _
_ _ _ _ _ X X _ X _
```

3 Discussion: heuristic and objective functions

Both of these types of functions give a value to a solution to evaluate how good it seems, but the `heuristic functions` require to know our goal in order to evaluate how appropriate a candidate seems to be. We know how an optimal solution looks like, so we can stop when we find it.

Whereas the `objective functions` evaluate a solution without knowing of the optimal solution. They just give an estimation on how good it seems. It allows us to compare two candidates in order to choose which one seems better (in order to reach our goal), but we can't know if we have an optimal solution.