TDT4136 - Exercise 3

Stein-Otto Svorstøl and Andreas Drivenes
3rd year MTDT

Fall 2014

1 Code

Handed in seperatly in the file EggCartonPuzzle.py. It's implemented in Python 3.4, and depends upon the external library numpy.

2 Key aspects involved in the SA-specialization

We chose to represent the board as a simple matrix, that is a list holding lists. True indicates an egg placed, False indicates an empty cell. We implemented in Python, so the implementation should be pretty straight forward. We had to set the start temperature pretty low for the algorithm to work well. Maybe this is because the generating of neighbours is quite random, and we don't have to pick a random neighbour over the "best" neighbour that often?

The objective function evaluates how good a solution may be. In our implementation it counts the number of eggs that breaks a constraint on each row, column and diagonal. It also estimates (this doesn't work well in general, but works for the puzzles given) the number of eggs that should be in the solution with min(n*m) * k. The constraints are then subtracted to the initial egg fraction score.

Neighbour generation is a very important part of this problem, as the tuning of the current state to these neighbours is what the whole solution is based upon. In our implementation we generate $N \cdot M$ neighbours. For each neighbour we take one random cell from the state and replace false/true. Everything else is the same. So we'll get a difference from the parent of 1/(n*m) in each neighbour. Some neighbours may be equal, so a possible improvement would be to implement it with a set instead of a list.

3 Diagrams for the EggCartonPuzzle

m = n = 5, k = 2

x	x	Egg	Egg	x
Egg	Egg	x	x	x
x	x	x	Egg	Egg
Egg	Egg	x	x	x
x	x	Egg	x	Egg

m = n = 6, k = 2

Egg	x	x	Egg	x	x
x	x	x	Egg	Egg	x
Egg	x	Egg	x	x	x
x	x	x	x	Egg	Egg
x	Egg	x	x	x	Egg
x	Egg	Egg	x	x	x

m = n = 8, k = 1. This is the 8 queen puzzle!

x	x	Egg	x	x	x	x	x
x	x	x	x	x	Egg	x	x
x	Egg	x	x	x	x	x	x
x	x	x	x	x	x	Egg	x
x	x	x	x	Egg	x	x	x
Egg	x	x	x	x	x	x	x
x	x	x	x	x	x	x	Egg
x	x	x	Egg	x	x	x	x

m = n = 10, k = 3

```
x   x   x   x   x   E   x   E   x   E

x   x   E   x   E   x   E   x   x   x

x   E   x   x   x   x   x   x   E   E

x   x   x   E   x   E   x   E   x   x

E   x   E   x   E   x   x   x   x   x

x   x   x   E   x   x   E   x   E   x

E   E   x   x   x   x   E   x   x   x

x   x   E   x   x   x   x   x   E   E

x   x   x   E   E   E   x   x   x   x

E   E   x   x   x   x   x   E   x   x
```

4 The heuristic vs the objective function

The objective function only evaluates how good one possible complete solution, a neighbour, may be. It only gives one a pointer on which neighbour may be the best, and does not really take our goal in mind. The heuristic function on the other hand, will evaluate how far we are from the goal, and use that as a pointer as to which direction in should go in. The heuristic function is based on a partial solution, we don't know the whole solution yet.