N-Queen game solved with

Genetic Algorithm and Hill Climbing

Ke Ding | CS420 | 05/12/2014

[kding@csupomona.edu](mailto:kding@csupomona.edu)

# Approach

## Class Structure

1. Class Queen represents the queen, it has a coordinate field.
2. Class Queens represents the state of the game, it’s a set of Queen Class.
3. Class Plain represents the node. It contains a Queens and an evaluation class to evaluate the state.
4. Interface Heuristic represents the evaluation, it take a current state as input, return the heuristic cost.
5. Interface Action represents the actions which take a state as input and return a new state.
6. Class HillClimbing represents the pure hill climbing algorithm based on above classes.
7. Class Genetic represents the genetic algorithm based on above classes.

## strategy

1. Sideways: In the pure hill climbing algorithm, allowed 30 sideways.
2. Selection: In the genetic algorithm, try Round Selection (select an individual according to its fitness value) and Binary Competition Selection (select two individuals randomly and chose the one with the larger fitness value), it turn out Binary Competition is better and faster.
3. Elite Strategy: take about 10 percentage of best individuals from the parent generation, put them into the children generation. Then sort the children generation according to their fitness value and return the best amount of individuals as the next generation, this amount is set as the population size form initialization. In this way I try to keep my best elites continue. But this strategy increases the possibility of local minimum.
4. Termination: use Catastrophe strategy as termination for genetic algorithm. It means that after detecting some amount of generations without fitness evaluation improvement, the program set a catastrophe that kill some specific percentage of elites which are the best of current generation. That is the strategy I used trying to escape the local minimum.
5. By Elite and Catastrophe strategy together, I try to get a balance between rapid convergence and avoiding local minimum.

# Experimental results

## Experiment Description

1. Run the n-queen program 5,000 times started with a random initial state for both hill climbing and genetic algorithm.
2. All the experiment codes are in the test.ReportTest unit test class.

## Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Search Cost | Run Time (milliseconds) | accuracy | # of Cases |
| Genetic | 2454 | 166 | 100% | 5000 |
| Hill Climbing | 34 | 12 | 13.35% | 5000 |

# Analysis

We can see that genetic algorithm takes more cost and run time but it can always get a perfect result.

On the other hand pure hill climbing algorithm is fast but only get a 13.35 accuracy.

# Extra finding

In general I think large population size is preferred by genetic algorithm. But in this N-Queen approach, I found that larger population such as thousands of individuals per generation is not very helpful for the result but increasing my program running time significantly. After adjust kinds of parameters, I choose the population size is equal to number of queens which help me get a 100 percentage accuracy result and very fast running time.