N queen Project Report

Name: Yueh-Lin Tsou ID: 012843142

1. Approach

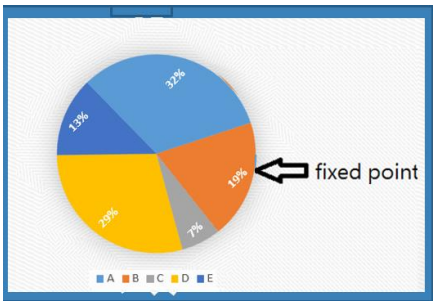
A. Genetic Algorithm

a. Fitness function:

The fitness function defines how good the state is, this program calculate the non-attacking value N * (N – 1) / 2 then minus 1 for each attacking pairs.

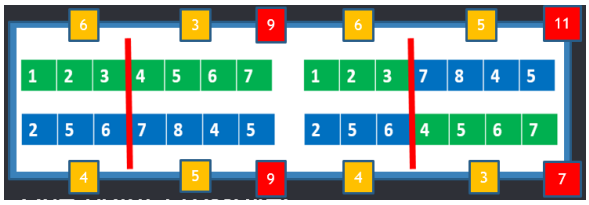
b. Parent Selection:

Every individual can become a parent with a probability which is proportional to its fitness. Therefore, the individuals which have a higher fitness value will have a higher chance of mating and propagating their features to the next generation. This process is like "**Roulette Wheel Selection**".



c. Crossover:

This program used ***one point crossover***, the process is like the figure below, randomly select a cut point then change part of the state solution.



d. Mutation:

It's a small random tweak in the state to get a new solution with a given probability. This program used ***insertion mutation*** which randomly select a queen then randomly change the position.

e. Survivor Selection :

The Survivor Selection determines which individuals are to be kicked out and which are to be kept in the next generation. This project generate 2N new population and sort them in descending order according to their fitness value, then select the top N population to be our next generation.

B. Simulated Annealing

a. Next state generator

Randomly select a queen then put in the random position to be our next possible state.

b. Selection function

If the fitness value of the next state is higher than the current state, then return the next state to be our new solution, otherwise, apply a probability to decide select or not

2. Comparison (Divided by cases for Avg.)

For the comparison, I generate around 1000 random cases for using both algorithm. Then output the solution probability and running time.

The **search cost** depend on how good or bad the initial state is and the random process, it also been restricted by the run limit, so the search cost is not an important point to be consider in local search algorithm.

Compare different between two algorithms

Run limit: 100,000 states

Genetic Algorithm

Cases	Population	Mutate Rate	Percentage	Running Time (ms)
1000	50	1/5	100 %	157304
	100	1/5	96.8	1112844

Simulated Annealing Algorithm

Cases	Temperature	Cooling factor	Percentage	Running Time (ms)
1000	10000	0.999	100 %	19019
		0.9	100 %	4288

Run limit: 10,000 states

Genetic Algorithm

Cases	Population	Mutate Rate	Percentage	Running Time (ms)
1000	50	1/5	93%	124050
	100	1/5	90.3	382867

Simulated Annealing Algorithm

Cases	Temperature	Cooling factor	Percentage	Running Time (ms)
1000	10000	0.999	100 %	17817
		0.9	100 %	4306

3. Analysis

   By comparing the data in the above table, we can see that:

   - Simulated Annealing Algorithm:
     Adjust “run limit” and “cooling factor”
   - Genetic Algorithm
     Adjust “run limit” and “population size”

     NOTE: I didn’t show the result for changing the mutate rate, because it took too much time to solve the problem, the program became harder to find the solution. Because if lower the mutate rate the solution will stock in the local maxima.

   Simulated annealing algorithm works very well for solving the N queen problem, it can solve almost 100% of the random problem. If decrease the cooling factor the program will solve the problem faster, but if the cooling factor is below than 0.7, the running time begin to increase and harder to find the solution. Because the program is too early to became not taking the bad solution to get away from the local maxima.

   Genetic algorithm also works well on solving the N queen problem, depend on the figure above we can see if increase the population size will extremely increase the program running time. Then if let the program run less state during the process, the percentage for solving the problem decrease to 90%. Because genetic algorithm is easy to stuck in the local maxima, it needs mutation function to help the program find the solution, so need more running state.

   Difficulties: Everything went fine. I gave lots of test print out to track the steps!

4. Finding

   (a) Find the next state for the current state:

   - Simulated Annealing Algorithm:
     - Select a random queen then put on a random position to be the next state

```
int Col = (int) (Math.random() * N);
int Row = 1 + (int) (Math.random() * N);
```

   - Genetic Algorithm:
     - Select two parents with some probability according to its fitness value
     - Select a random cut point for crossover
     - Do the mutation with a given probability
     - Return next state
     - Finally, do survivor selection for being our next generation