N-Queens Problem Using Genetic Algorithm

Genetic Algorithm

- Genetic Algorithm are non-deterministic algorithm that follows a natural order to determine the best possible solution of a problem. In our program the genetic algorithm will be used in order to determine the conflicts between any two queens on the chess board and will assign the fitness of the pattern according to the conflict.
- The fittest patterns along with their offspring's will pass on to the next generation and conflicts will be calculated accordingly. To eliminate any possibility of biased result, we have added mutations in our code so that a few mutated patterns are added after fixed intervals. This would ensure that the results that we get are not biased to the parents that are generated.

Problem Statement

- The N-Queens problem is a classic problem that is used to determine the position of queens that can be spread on the chess board such that no two queens can kill each other. The problem can be solved in many possible ways including backtracking, branch and bound, etc.
  Here in our project we try to solve N-Queens using Genetic algorithm. The general running time of N queens problem is O(n!) and hence we have tried to reduce this using our genetic algorithm approach.

Approach

- We followed the approach of solving the N-Queens by implementing chromosomes which have maximum length, fitness, conflicts.

- Firstly, we plotted the queens on the diagonals and then checked all four directions to make sure that there are no conflicts. The fitness is calculated based on the conflicts.

- Based on the fitness, parent with highest fitness will come first, followed by the next in order and so on.

- For mating, we have used cross over partial mapping.

- **Mutation:** If the queen in the parent has same position in the two boards, same position will be given to the children in the new board. But if the position of parents in both boards are different, child would be given a random value in the new board.

- Using Roulette wheel for selecting parent starting from a fixed point and then spinning the wheel. The region which comes in the front of the fixed point is chosen as the parent. And the process is repeated for the next iteration.

- **Genotype** in genetic algorithm corresponds to chromosome in our case(heritable aspect)

- **Phenotype** is defined as the number of conflicts between queens (the trait by which two queens can attack each other).

- **Fitness** which is calculated based on the number of conflicts of the queens. So if there are maximum conflicts, fitness will be lowest and vice versa.

- Our **test cases** include test cases for fitness, calculation of offspring's.

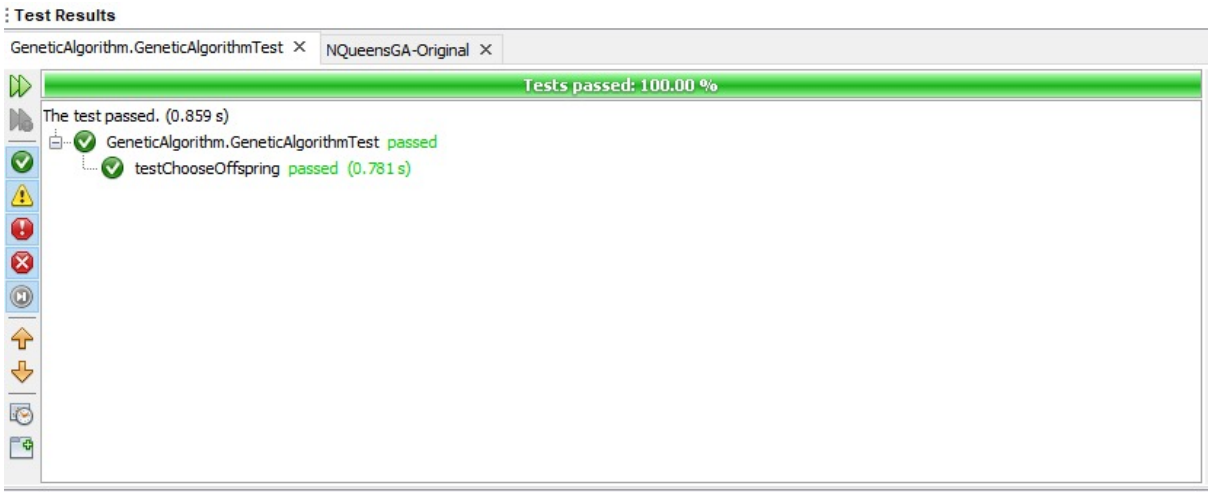
Implementation Details:

1. We have implemented genetic algorithm separately which encompasses the genetic code and also generates the random mutation.
2. **Gene expression:** We had three fixed criteria to find the fitness function. The position of the queen should be such that no other queen is present horizontally, vertically and diagonally. Such a criteria enabled us to find the number of conflicts (fitness function).
3. **Fitness function:** The above criteria enabled us to find the number of conflicts. The more the number of conflicts, fitness will be the lowest and vice versa.

Results:

Unit Test Cases:

Test cases for offspring's:



```
Genetic Algorithm
Parameters
Maximum Length divided by Number 8
Starting Population of chromosomes: 40
Maximum  epochs: 3000
Probability of mating of chromosome: 0.7
Rate of Mutation between chromosomes: 0.001
Minimum number of parent selected: 10
Maximum number of parent selected: 30
Offspring generated per generation: 20
Number of minimum shuffle done: 8
Number of maximum shuffle done: 20

Cycle: 1
Running Time: 30312000
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . .
. . . . . X . .
. . . . . X
. . X . . . . .
. . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 2
Running Time: 16198700
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . .
. . . . . X . .
. . . . . X
. . X . . . . .
. . . . . X .
. . . X . . . .
```

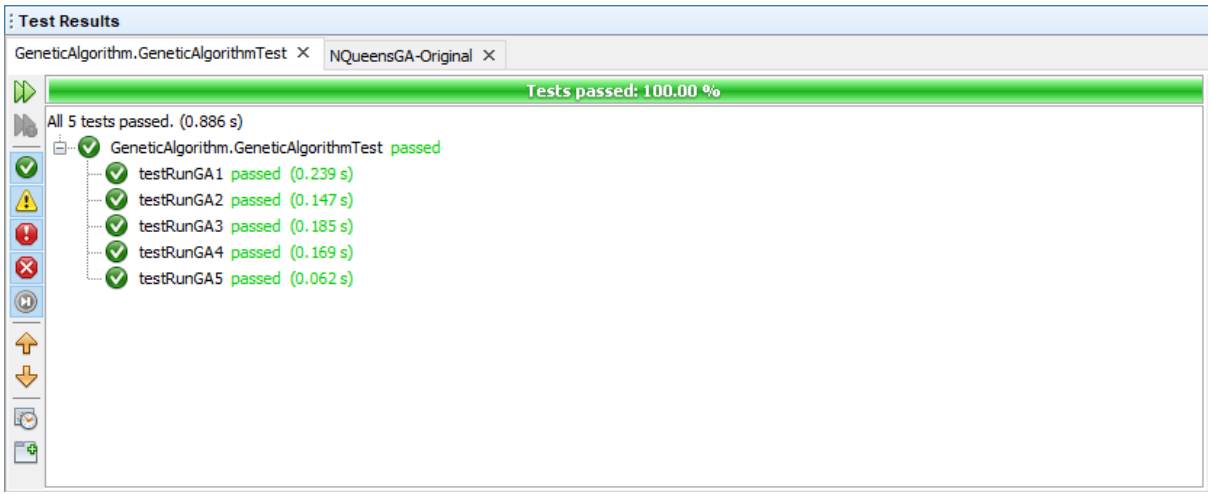
```
. . X . . . . .
. . . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 5
Running Time: 13496700
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . X . .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 6
Running Time: 10395100
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . X . .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 7
Running Time: 70972300
Target of Epoch hit: 22
The new population size generated: 640
```

```
643
644  Cycle: 46
645  Running Time: 83425300
646  Target of Epoch hit: 22
647  The new population size generated: 640
648
649  X . . . . . . .
650  . . . . . X . .
651  . . . . . . . X
652  . . X . . . . .
653  . . . . . . X .
654  . . . X . . . .
655  . X . . . . . .
656  . . . . X . . .
657
658  Cycle: 47
659  Running Time: 3363100
660  Target of Epoch hit: 22
661  The new population size generated: 640
662
663  X . . . . . . .
664  . . . . . X . .
665  . . . . . . . X
666  . . X . . . . .
667  . . . . . . X .
668  . . . X . . . .
669  . X . . . . . .
670  . . . . X . . .
671
672  Cycle: 48
673  Running Time: 3963000
674  Target of Epoch hit: 22
675  The new population size generated: 640
676
677  X . . . . . . .
678  . . . . . X . .
679  . . . . . . . X
680  . . X . . . . .
```

```
Cycle: 50
Running Time: 3748400
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Summary

30312000
16198700
10748500
12758600
13496700
10395100
70972300
22504300
11413700
7564800
5218500
5675100
83995100
16253200
17738700
7666900
5558700
3543600
```

```
714  Summary
715
716  30312000
717  16198700
718  10748500
719  12758600
720  13496700
721  10395100
722  70972300
723  22504300
724  11413700
725  7564800
726  5218500
727  5675100
728  83995100
729  16253200
730  17738700
731  7666900
732  5558700
733  3543600
734  3254700
735  87385500
736  4941200
737  4880400
738  3913000
739  3601600
740  4028500
741  3788900
742  82306300
743  3639100
744  3557700
745  3651600
746  3897000
747  3748100
748  3593400
749  3896900
750  4292300
```

Test cases results when running genetic algorithm:



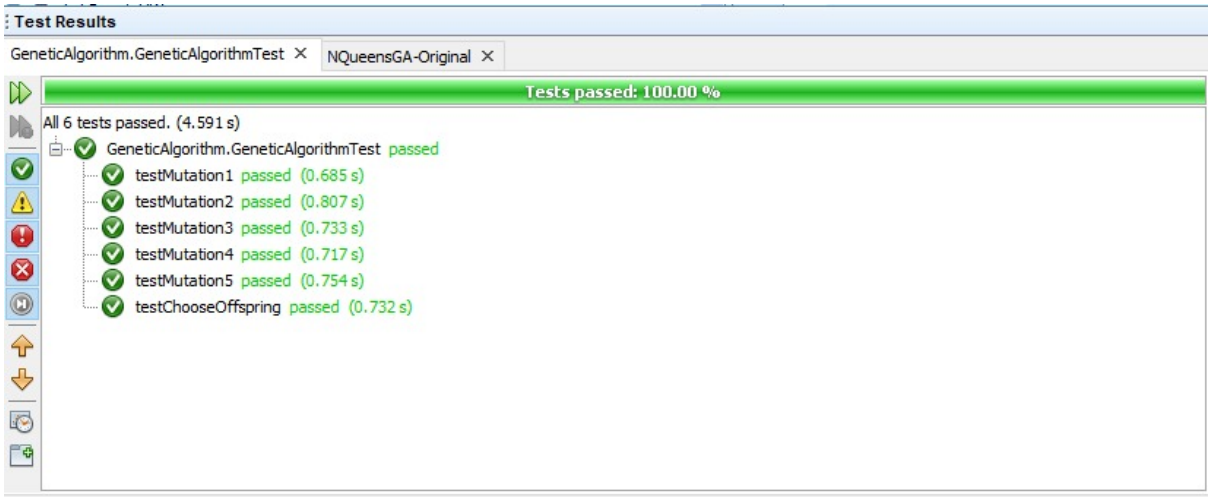
```
Genetic Algorithm
Parameters
Maximum Length divided by Number 4
Starting Population of chromosomes: 40
Maximum  epochs: 15
Probability of mating of chromosome: 0.7
Rate of Mutation between chromosomes: 0.001
Minimum number of parent selected: 10
Maximum number of parent selected: 30
Offspring generated per generation: 20
Number of minimum shuffle done: 8
Number of maximum shuffle done: 20

Cycle: 1
Running Time: 740100
Target of Epoch hit: 1
The new population size generated: 66

. X . .
. . . X
X . . .
. . X .

. . X .
X . . .
. . . X
. X . .

. . X .
X . . .
. . . X
. X . .

. . X .
X . . .
. . . X
. X . .
```

```
1207  . . X .
1208
1209  Cycle: 41
1210  Running Time: 561200
1211  Target of Epoch hit: 1
1212  The new population size generated: 72
1213
1214  . . X .
1215  X . . .
1216  . . . X
1217  . X . .
1218
1219  . X . .
1220  . . . X
1221  X . . .
1222  . . X .
1223
1224  . . X .
1225  X . . .
1226  . . . X
1227  . X . .
1228
1229  . . X .
1230  X . . .
1231  . . . X
1232  . X . .
1233
1234  Cycle: 42
1235  Running Time: 705800
1236  Target of Epoch hit: 1
1237  The new population size generated: 70
1238
1239  . . X .
1240  X . . .
1241  . . . X
1242  . X . .
1243
1244  . X . .
```

```
1499  Cycle: 50
1500  Running Time: 491000
1501  Target of Epoch hit: 1
1502  The new population size generated: 70
1503
1504  . . X .
1505  X . . .
1506  . . . X
1507  . X . .
1508
1509  . X . .
1510  . . . X
1511  X . . .
1512  . . X .
1513
1514  . X . .
1515  . . . X
1516  X . . .
1517  . . X .
1518
1519  . . X .
1520  X . . .
1521  . . . X
1522  . X . .
1523
1524  . X . .
1525  . . . X
1526  X . . .
1527  . . X .
1528
1529  . . X .
1530  X . . .
1531  . . . X
1532  . X . .
1533
1534  . . X .
1535  X . . .
1536  . . . X
```

1539	Summary
1540	
1541	740100
1542	511100
1543	504200
1544	543600
1545	492400
1546	90176500
1547	1376000
1548	1349400
1549	1639700
1550	1195200
1551	1400700
1552	772100
1553	1044800
1554	3063300
1555	450300
1556	741000
1557	1025200
1558	3272900
1559	475400
1560	440200
1561	393500
1562	253800
1563	303300
1564	381700
1565	583800
1566	458100
1567	699200
1568	322000
1569	1074000
1570	828900
1571	9074100
1572	508800
1573	478100
1574	705400
1575	878000
1576	378400

Test cases for mutation:




```
Genetic Algorithm
Parameters
Maximum Length divided by Number 8
Starting Population of chromosomes: 40
Maximum  epochs: 3000
Probability of mating of chromosome: 0.7
Rate of Mutation between chromosomes: 0.001
Minimum number of parent selected: 10
Maximum number of parent selected: 30
Offspring generated per generation: 20
Number of minimum shuffle done: 8
Number of maximum shuffle done: 20

Cycle: 1
Running Time: 9964600
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 2
Running Time: 8617600
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . . X .
. . . X . . . .
```

```
. . . . X . . .

Cycle: 48
Running Time: 4143900
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 49
Running Time: 4052600
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
. . X . . . . .
. . . . . . X .
. . . X . . . .
. X . . . . . .
. . . . X . . .

Cycle: 50
Running Time: 3849300
Target of Epoch hit: 22
The new population size generated: 640

X . . . . . . .
. . . . . X . .
. . . . . . . X
```

```
714  Summary
715
716  9964600
717  8617600
718  5952200
719  4177900
720  3030400
721  79202200
722  3168200
723  3110300
724  2155700
725  2432900
726  2286500
727  2335800
728  80739200
729  6050900
730  5706400
731  5024800
732  4164500
733  4030500
734  4308200
735  72188400
736  7642800
737  9642000
738  11888600
739  10580400
740  8020000
741  4724700
742  3724600
743  3078200
744  3042800
745  3323800
746  88640400
747  6766600
748  6119500
749  4697100
750  4682500
751  5285700
```

Normal text file

Conclusion:

In this we have enacted the N queens problem using genetic algorithm, a non-deterministic algorithm and have documented our results.

References:

Genetic Algorithm:
Genetic Algorithm Introdcution
Creating a Genetic Algorithm Project
Tutorial on Genetic Algorithm

N queens Algorithm:
Understanding N queens Using Backtracking Algorithm
Improving N queens using Branch and Bound Method
Java Code for N queens Algorithm Using Genetic Algorithm