CS-4110 Assignment2
Prof. Bolufe-Rohler

QTB problem with Genetic Algorithm

Yuki Yamada
138557

USER INSTRUCTIONS:
You can run my program from Solver.java class. I wrote main method in Solver, so you can run main method too.

DESCRIPTION:
Classes:

GeneticAlgorithm.java

GASelector.java
GACrossover.java
GAMutator.java

QTBPopulation.java
QTBPopulationFactory.java
QTBValidator
QTBPrinter

GeneticAlgorithm class implements SearchAlgorithm interface and overrides Start() method. GeneticAlgorithm has GASelector, GACrossover and GAMutator objects and uses them for genetic algorithm.

GASelector class has selection methods. Roulette Wheel Selection is used.
GACrossover class has crossover methods. Two points are selected and crossover the between.
GAMutator class has mutating methods. Mutation occurs 1/5 possibility.

QTBPopulation denotes a state (a board with all the pieces).
QTBPopulationFactory creates an initial population.

QTBPrinter is a debugging class. (Not used for testing).

ANALYSIS:

There are some factors which affect the result.
1. # of populations in one generation
2. possibility of mutation to occur

I made one generation to be 100 populations. With 10,000 evaluations, approximately the 100th generation (10,000 / 100) will be created. With 1,000,000 evaluations,

approximately 10,000th generation (1,000,000 / 100) will be created. More initial population means more initially random states are created. When you are stopped by a local minimum, it is a good idea to create new, random populations. It is a trade off of having more offsprings or refreshing with new, random populations.

Possibility of mutation occurring is also important. I set it so that 20% of populations gets mutated. If you mutate two many populations in one generation, you will ruin good populations. Especially, when you have less number of available evaluations, too many mutation will ruin creating good offsprings.

My Genetic Algorithm set

One generation = 100 populations (boards)
Possibility of mutation happening = 20%

RESULTS:

#	Inputs	Genetic Algorithm	Semi-Random
1	**Eval = 10,000** **board size = 20 x 20** **Queen = 10** **Tower = 7** **Bishop = 5**	100 times Ave: 5.77 Min: 2 Max: 8	100 times Ave: 7.28 Min: 4 Max: 10
2	**Eval = 10,000,000** **board size = 20 x 20** **Queen = 10** **Tower = 7** **Bishop = 5**	100 times Ave: 1.77 Min: 0 Max: 4	100 times Ave: 2.59 Min: 0 Max: 6
3	**Eval = 1,000,000** **board size = 100 x 100** **Queen = 20** **Tower = 20** **Bishop = 20**	3 times Ave: 0 Min: 0 Max: 0	3 times Ave: 0 Min: 0 Max: 0
4	**Eval = 1,000,000,** **board size = 100 x 100** **Queen = 60** **Tower = 20** **Bishop = 60**	3 times Ave: 31.3 Min: 27 Max: 34	3 times Ave: 58.3 Min: 56 Max: 62

#1 This one is difficult to get good fitness values because with a generation with 100 populations, only 100th offspring generation is created.

Difficulty is #2 > #3 since
#2 20x20= 400 squares with total 22 pieces (22/400 on board)
#3 100x100 = 10,000 squares with total 60 pieces (60/10,000 on board)

#2 You have more evaluations, so my algorithm can get fitness value = 0.
#3 This one is easy to get fitness value = 0.

#4 The best fitness values are around f=27 compared with the semi-random algorithm f=50.

Selection
- Roulette Wheel Selection (Adopted)
- Rank-Based Selection

Crossover
- One point
- Two points (Adopted)
- Randomly choose crossover indexes

The input #2 and #4 takes much longer to proceed than the others because the #pieces / #square on board is relatively high. This results in having more invalid offsprings after each reproductive iteration. Since invalid population does not consume the number of evaluation with objective function, it takes longer to complete the task.

CONCLUSION:
Depending on how many evaluations are available, we should adapt the number of population, the mutation percentages, and selection method. The more difficult a given problem is, the better Genetic Algorithm behaves. With small number or easy problems, it becomes harder to see how good Genetic Algorithm is.

BEST VALID POPULATIONS (Genetic Algorithm)
maxEvals = 1000000, n = 20, nQ = 10, nT = 7, nB = 5
Fitness = 0
x = {8, 4, 1,20, 5,18, 7,19,14,16,12, 3,11, 6, 9,17,10,18,13,13, 2, 2}
y = {1,15, 9, 2,10, 8,11,17,16,12, 6,13, 3,18, 4,14, 7, 5,19,14,19, 5}
T = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2}

maxEvals = 1000000, n = 100, nQ = 20, nT = 20, nB = 20 (Fitness = 0)
x = {51,16,87,77,19, 2,39,69,53,49,55,23,60, 6,38,12,22,
3,81,88,57,64,32,46,39,75,84,73,65,41, 8,91,50,66,
1,37,97,71,97,24,33,26,15,54,88,34,18,11,44,25,85,59,64,10,98,94,97,86,92,83,13,88,93,
9,76,18,18,39,68, 7,99,24,35,70,52,45,28, 5,45,48, 4,74,90,56,72,
4,61,93,62,14,78,89,36,96,29,95,90,
4,27,89,80,43,21,79,78,27,82,63,20,79,31,95,95,14,14,
4,79,47,61,62,62,58,78,96,56,31,27,58,80,17,14,40,90,100,43,67,95,95,43,89}

y = {32,13,65,93, 2,60,68,90,66,64,89,76,28,98,85,69,45,21,95,19,81,24,99,86,93,23,77,
5,71,67,97,78,40,31,79,33,83,63,86,40,
9,20,35,91,86,14,27,92,46,26,57,47,46,58,49,22,40,96,51,27,72,40,48,73,
1,19,28,46,100,42,10,27, 6,54,40,86,61,38,27,11,59,53,36,84,94,12,15,95,
3,59,34,41,55,62, 4,59,43, 8,87,59,80, 4,94, 8, 8,34,18,12,37,41,94,62,
7,39,44,50,16,88,88, 4,39,16,18,43,94,13,12,70,75, 8,17,82, 4,18,17,12,34,16,74,88}

T = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}

maxEvals = 1000000, n = 100, nQ = 20, nT = 20, nB = 20 (Fitness = 26)
x = {68,30,76,32,91,33,90,73, 2,74, 1,16,
5,36,61,20,79,82,95,12,96,11,93,46,59,58,17,49,99,19,16,45,81,91,53, 5, 7,15,65,94,
9,100,80,39,87, 3,37,44,17,62,41,78,47,35,77,34,66,50,92,23,48,25,
5,88,83,49,38,50,29,72,
8,65,22,88,24,54,84,89,18,71,97,31,52,60,86,98,27,42,10,60,51,21,26,52,55,51,86,
6,13,31,60,55,97,28,86, 6,67,69,28,27, 6,13,27,63,52,40,
6,63,85,75,69,97,75,10,14,14,85,43,40,27,40,27,70,51, 4,97,98,64,86,98}

y = {47,88, 1,28,10,
2,51,79,62,66,74,85,53,60,53,73,27,68,98,61,13,76,30,84,17,11,84,85,50,49,10,40,57,37,
31,20,46, 6,36,93,52,42,18,12,95,69,44,75,35,16,81,80,15,53,54,48,59,85,41,43,22, 8,58,
5,10,37,94,66,91,99,53,39,24,53,86,82,67,56,36,100,78,63,96, 7,87,64,90,14,87,97,
3,34,87,64,71,92,83,77,89,19, 3,25,87,96,21, 4, 7,34, 3,72,65,92,38,96,78,87,23,97,
7,64,26,77,97,33,23,97,19, 3,97,32, 4, 9, 9,38,23, 3,14,26,90,83}

T = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}