Project 1

Knut Halvor Skrede

February 11, 2011

Part A)

Code description

The code for the EA is written in c++. It consists of 4 classes; BasicEA, Strategies, Population and Individual. The BasicEA class implements the main loop. This loop goes through the main functionality of the Population class. The main functionality of the population class is to call the member functions implemented in the Individual class using the strategies

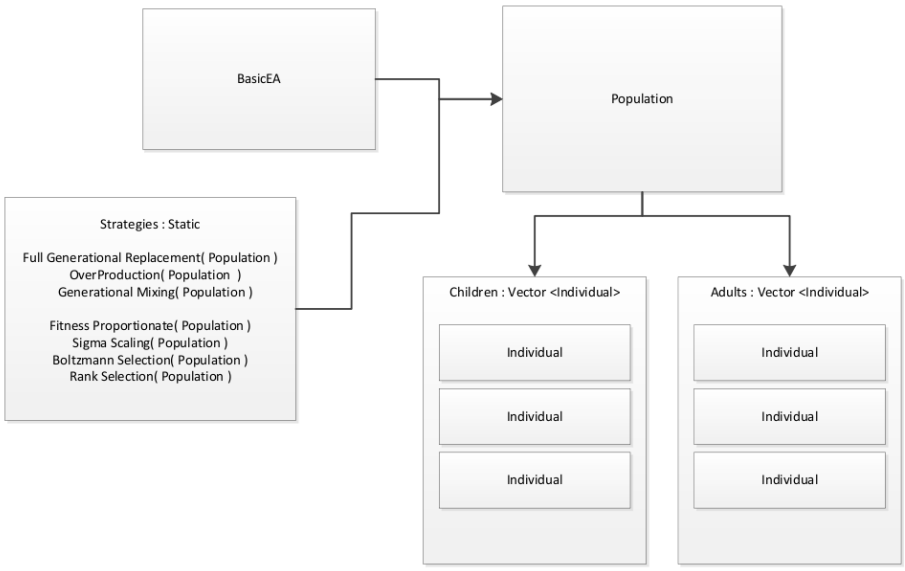

Figure 1: Simple layout of classes

implemented in the Strategies class. See figure 1 for a simple layout. The Population class also implements functionality to collect statistics and other helpfull information.

The program takes the following parameters as arguments:

1. Size of child pool [integer]
2. Size of adult pool [integer]
3. number of generations [integer]
4. mutation rate [float between 0.0 and 1.0]
5. crossover rate [float between 0.0 and 1.0]
6. selection protocol [integer between 1 and 3]
   (1) Full Generational replacement (parameter 1 and 2 must be equal)
   (2) Over-production (parameter 1 must be larger than paramter 2)
   (3) Generational-mixing
7. selection strategy [integer between 1 and 4]
   (1) Fitness proportionate
   (2) Sigma scaling
   (3) Boltzmann selection
   (4) Rank selection
   (5) Tournament selection
8. Elitism [integer]
9. If boltzmann selection: Temperature [float greater than 0]
   If tournament selection: Tournament size [integer greater than 2 and less than halv the size of adult pool]

Genotype representation

The genotype is represented as an array of 40 boolean values.

Developmental method

The developmental method converts the genotype (array of 40 boolean values) into a phenotype (array of 10 integers between 0 and 15.)

Fitness evaluation

The Individuals class implements it's own fitness evaluation function. Here it is simply the sum of all integers in the phenotype.

Selection protocols

The selection mechanisms for parent selection fills the child pool with a number of children, provided as an argument to the application. The selection protocols are then responsible for choosing which of the children become adults. All the suggested selection protocols (Full generational replacement, Over-production and Generational Mixing) are implemented in the Strategies class.

Selection mechanisms

Tournament selection

When using tournament selection, and a crossover is to be performed, two tournaments are arranged to find both parents. Therefore, the size of the tournament groups must be half of the size of the adult pool.

Other selection mechanisms

All the other selection mechanisms are implemented as a roulette wheel, where the expected values are calculated according to the respective methods (Fitness proportionate, Sigma scaling, Boltzmann selection or Rank selection.)

Genetic operators

Crossover

The crossover functionality was implemented as a random bitmask that chooses which genotypes come from each parent. And by remembering that

bitmask, and by always producing two children, It is guaranteed that the crossover will fully represent it's parents genotypes in the next generation. This seemed more natural than to simply take half from one parent and half from the other.

Mutation

Mutation is implemented as a function that flips a random bit in the genotype. All children has a probability equal to the mutation rate parameter to be mutated.

Elitism

Elitism is implemented such that a parameter (elitism) chooses how many of the best individuals are to be cloned into the child pool. However, elite individuals have to be selected according to the selection mechanism in order to become adults, and since I mutate across the entire child pool after reproduction, elite individuals may be mutated.

Code modularity

All the main strategies and protocols are implemented in the Strategies class and uses the functions implemented in the Population and Individual classes. This way, I just have to change the Individual class to adapt problems to the BasicEA and I only have to alter the Strategies class to alter the main strategies and protocols.

Performance analysis

The minimal population that was found to consistently solve the 40 bit one-max problem was 7. The program was then run with the parameters: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Fitness proportionate, elitism: 2. See figure 2 for a plotting of the EA run.

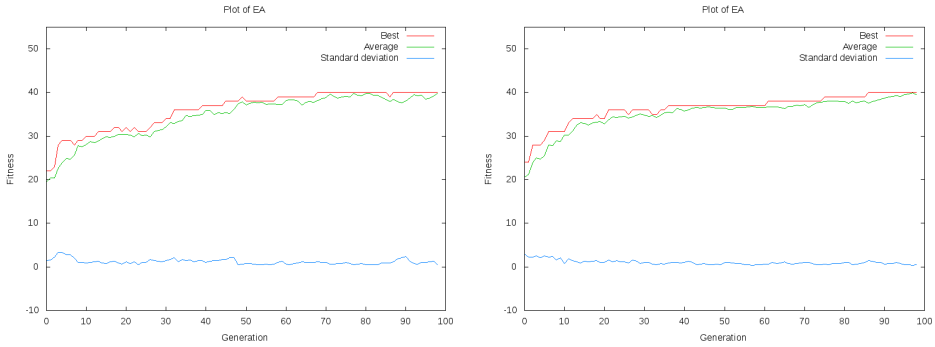

Figure 2: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Fitness proportionate, elitism: 2

Changing the mutation rate to 0.1 and the crossover rate to 0.9 resulted in no solution being found within 100 generations. See figure 3.

Changing the mutation rate to 0.6 and the crossover rate to 0.3 resulted in a solution beeing found (See figure 4,) but not as early and not as consistently.

Changing the mutation rate to 0.8 and the crossover rate to 0.2 resulted in a solution beeing found (See figure 5,) but not as early and not as consistently.

Parent selection experiment

Sigma scaling clearly cause the population to converge faster. See figure 6.

Boltzmann scaling does not scale as fast as sigma scaling, but faster than fitness proportionate. See figure 7.

Rank selection performed approximately as fitness proportionate. See figure 8.

Tournament selection performed well. But not as good as sigma scaling. See figure 9.

The selection mechanism that found the solution in the least amount of generations and found it consistently, was Sigma Scaling. Sigma scaling is

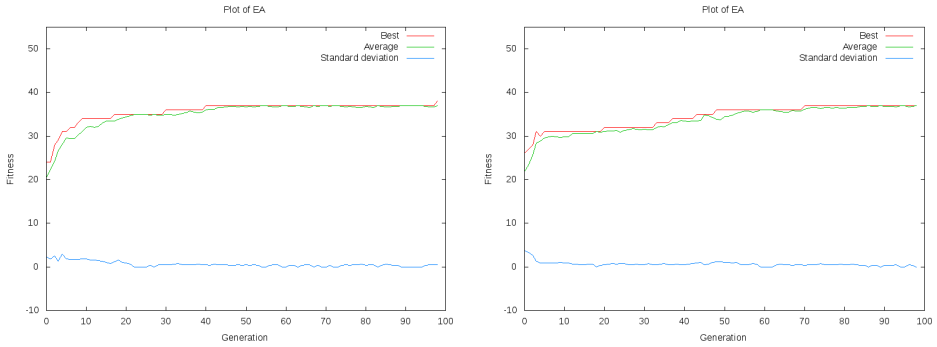

Figure 3: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.1, crossover rate: 0.9, selection protocol: Full generational replacement, selection strategy: Fitness proportionate, elitism: 2

clearly the best of our choices for solving the one-max problem.

Random bit vector

The Indivudal class was altered to develop a phenotype of 10 integers where each integer represented how many of the genotypes 40 bits where equal to the array: [1,0,0,0,1,1,0,1,0,1,1,0,0,0,1,1,1,1,0,0,0,1,1,0,1,0,1,0,0,0,0,0,1,1,0,1,0,1,1,0] in an interval from n to $n + 4$.

The fitness evaluation function remained the same, summing the integers representing the phenotype.

Expected result

Changing the problem to find a random bit-vector should not increase the difficulty of the problem as the search-space for the evolutionary algorithm is the same and still has only one correct solution. Therefore, adjusting the developmental method should be enough to find the correct solution using the same parameters as the ones that found the solution to the one-max problem.

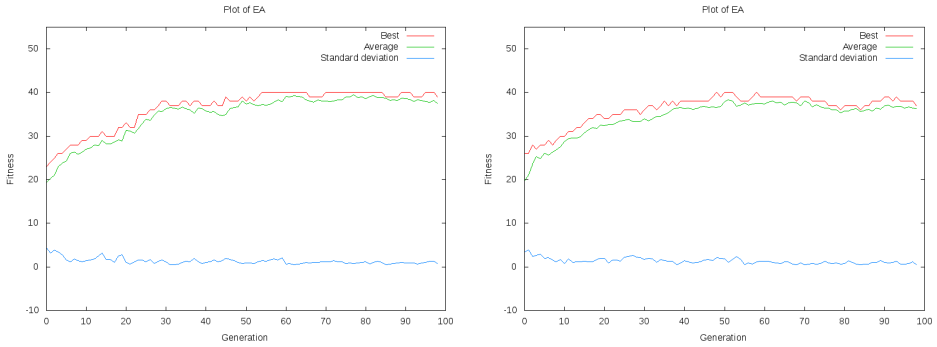

Figure 4: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.6, crossover rate: 0.3, selection protocol: Full generational replacement, selection strategy: Fitness proportionate, elitism: 2

Actual result

Running the EA with the same parameters as those that consistently solved the one-max problem with a minimal population size, but with different selection mechanisms, resulted in very similar plots. This means that we got the result we expected.

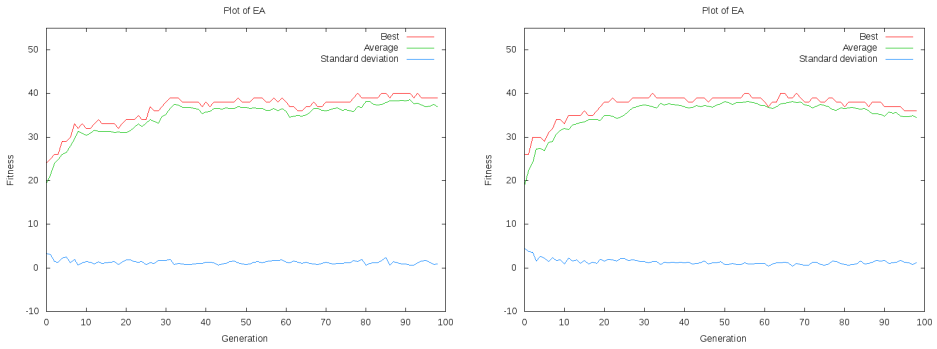

Figure 5: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.8, crossover rate: 0.2, selection protocol: Full generational replacement, selection strategy: Fitness proportionate, elitism: 2

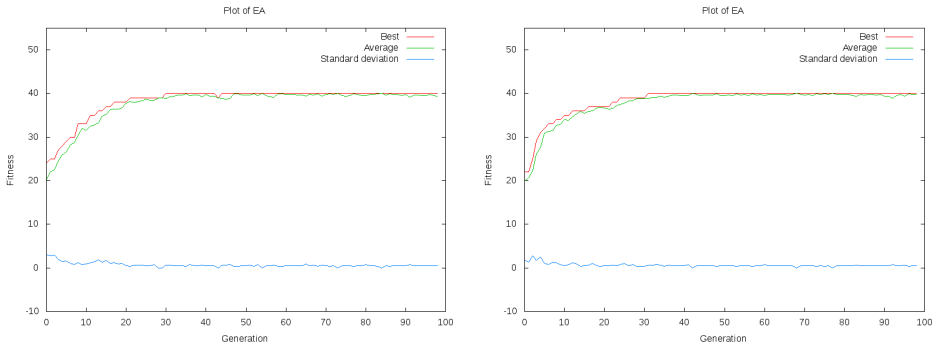

Figure 6: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Sigma Scaling, elitism: 2

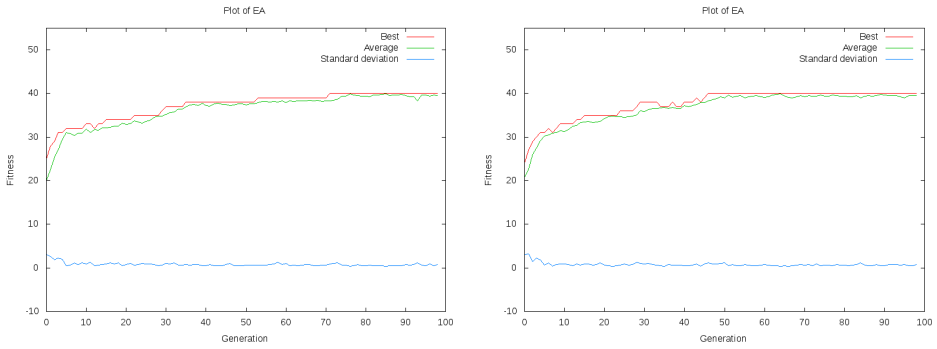

Figure 7: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Boltzmann selection, elitism: 2, Temperature: 2

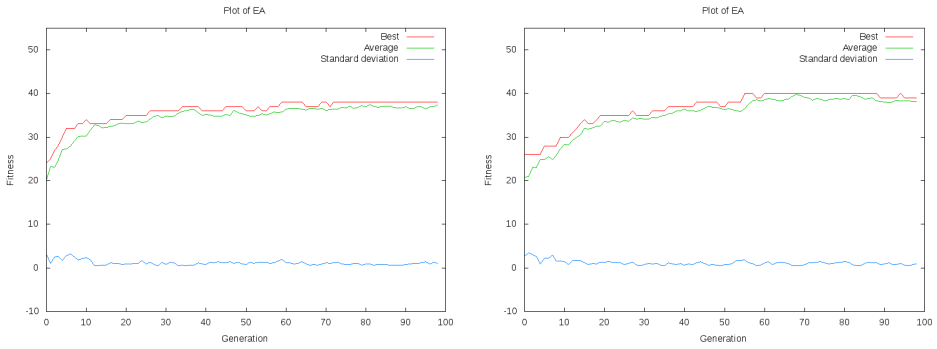

Figure 8: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Rank Selection, elitism: 2

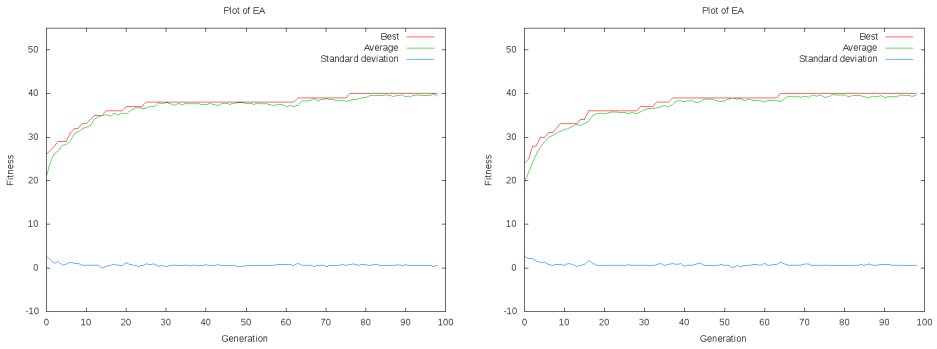

Figure 9: Plotting of two EA runs on the one-max problem: Size of child pool: 7, Size of adult pool: 7, number of generations: 99, mutation rate: 0.3, crossover rate: 0.6, selection protocol: Full generational replacement, selection strategy: Tournament selection, elitism: 2, Tournament size: 3