ECTA Homework 4
Multiobjective Optimization with the Non-dominated Sorting Genetic Algorithm II

Debaraj Barua (9030412), Md Zahiduzzaman (9030432)

June 28, 2018

1 Assignment Description

1. Implement the NSGA-II algorithm and apply it to a toy problem
   - Bit string with length 20
   - Maximize the number of leading zeros (zeros in a row at the front)
   - Maximize the number of trailing ones (ones in a row at the back)
2. Show that your algorithm works by plotting the population at various stage of the algorithm

2 Submission Instructions

Follow along with the instructions in this PDF, filling in your own code, data, and observations as noted. Your own data should be inserted into the latex code of the PDF and recompiled. All code must be done in MATLAB.

To be perfectly clear we expect two submissions to LEA:

1. 1 PDF (report) – a modified version of your submission PDF, with your own code snippets, figures, and responses inserted
2. 1 ZIP (code and data) – a .zip file containing all code use to run experiments (.m files) *and* resulting data as a .mat file
3. 1 GIF (algorithm progress) – use the file on the MATLAB file exchange: `https://www.mathworks.com/matlabcentral/fileexchange/63239-gif`

3 The Assignment

3.1 NSGA-II (75pts)

- (50pts) Implement NSGA-II to find all non-dominated solutions to the trailing ones, leading zeros problem.
  - Bitstring with length 20
  - Population size of 100
  - Generations 100
  - Hints:
    - Crossover and mutation can be performed just as in other bit string problems, e.g. one-max
    - The `sortrows` function can be used to sort matrices, you can use this first before implementing NSGAs sorting
- (20pts) Visualize the progress of your algorithm over a single 100 generation run with an animated gif (1 frame every generation).
  - Use the code here: `https://www.mathworks.com/matlabcentral/fileexchange/63239-gif` to create gif
    - Set the timing so that the gif completes in a reasonable amount of time (between 10 and 20 seconds)
  - Fronts can be visualized with the code snippets attached (`displayFronts.m`)

  The visualization is shown in plot.gif and submitted with the assignment
- (5pts) At each iteration mark the individuals which carry on to the next population, and which do not (you will have to code this yourself).
  Individuals carried to the next population is marked as green circle and others are marked as red circle. The individuals at the first front are marked as filled circle

3.2 Short Answer (25pts)

- (10pts) Compare the sort used by NSGA-II with a variety of population sizes. How long does 100 generations take with each approach when using a population size of:

  1. 10: 25.6265 seconds, does not find any of the pareto optimal solution.
  2. 100: 50.1688 seconds, finds all 21 pareto optimal solutions.
  3. 1000: 1828.7204 seconds, The population size is too high because with genome of length 20 there can be only 210 unique solutions ($\frac{N\times(N+1)}{2}$) and population size of larger then this is not helpful.

- (5pts) Plot the end result of a single run with [100 pop and 100 gen] and [10 pop and 1000 gen]. Describe the difference between the end results. Which is preferable?

  Legend for graphs:

  - The levels start at 1 for the right most points (in blue) and increases towards left.
  - Any filled inner circle represents the individuals in first level.
  - A green empty inner circle represents the individuals carried to the next population.
  - A red empty inner circle represents individuals not moving to the next generation.

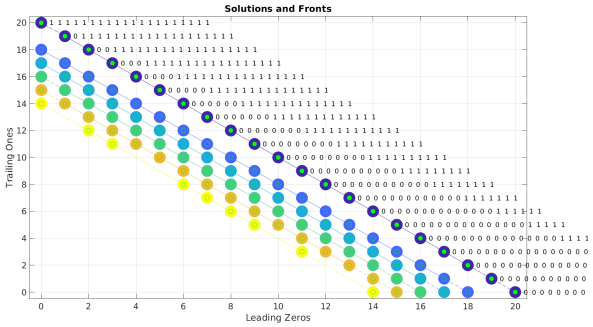

Figure 1: pop 100, gen 100, takes 53.5034 seconds

The end result shows that due to the small population size in case of population size 10, it does not find all 21 pareto optimal solutions and it takes longer to run. So, pop 100 and gen 100 is better.

Figure 2: pop 10, gen 1000, takes 233.6489 seconds

- (5pts) Imagine you were to replace the objective of "leading zeros" with "largest binary number". Predict the result, and give your reasoning.
  The highest binary number in this range is $2^{20} - 1$, this can be reached by having twenty "1"s. This solution satisfies both the conditions of maximizing trailing ones and getting the largest binary number. Thus, we can get only one pareto optimal solution.

- (5pts) Imagine you were to add a third objective: "non-consecutive ones and zeros" (ones not touching ones and zeros not touching zeros, e.g. 0101 and 1010 are the most optimal 4 bit solutions). How would you adjust the hyperparameters to get a satisfactory result?
  The search space for this problem will increase as we have one more objective to fulfill. As such, we would, intuitively need to increase the generation and population size.

  To select the hyper parameters for the modified implementation, we run the process multiple times for different combinations of population size and generations.

  - For 200 population and 100 generation, 116 non-dominated solutions are found.
  - For 300 population and 100 generation, 115 non-dominated solutions are found.
  - For 200 population and 200 generation, 120 non-dominated solutions are found.
  - For 200 population and 300 generation, 120 non-dominated solutions are found.

  We observed that the combination of 200 population and 200 generation gives the best result.

- (5pts) In many GAs and ESs populations must be ranked, but no special methods are used. Why is a faster sort in MOO so important?

A faster sort is important because of the the number of comparison performed and less comparison requires less time and vice versa. The approach we followed requires $\mathcal{O}(MN^2)$ whereas a naive implementation is also possible which is easier to implement but requires $\mathcal{O}(MN^3)$ comparison operations. Here M is number of objectives and N is the population size. It makes a huge difference for large population size.

3.3 ** Extra Credit ** (+ 10pts in examination)

Implement the third objective "non-consecutive ones and zeros"

1. How many non-dominated solutions are possible? (Hint: start with a smaller length and test)

   Maximum 120 non-dominated solutions have been found for 200 generation and 200 population.

2. What changes did you make to the algorithm and hyperparameters to get a good result?

   - The hyper parameters have been updated to get a better result.
   - The population size and generation was selected as 200 each.
   - We changed only the function that returns objective values for an individual. Hence, the function returns three values instead of two for each individual in the population.
   - The calculation of levels and crowding distance did not require any change because our original algorithm was implemented in a generic way and the number of objectives is parameterized and levels and crowding distances are calculated accordingly.

3. List the solutions in your 1st front. Are they all Pareto optimal? How complete is your front (in percentage, based on #1)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1
0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1
0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Table 1: Sample solutions in the first front

We found 120 solutions in the first front which are all Pareto optimal. A sample of 10 solutions have been shown above, the remaining solutions have been provided in a mat file (first_front.mat).

4. Plot the end result in 3D. (Use `plot3` or `scatter3`)

Legend for 3D graph:

- The levels start at 1 for the top most points (in blue) and increases towards bottom.
- Any filled inner circle represents the individuals in first level.
- A green empty inner circle represents the individuals carried to the next population.
- A red empty inner circle represents individuals not moving to the next generation.

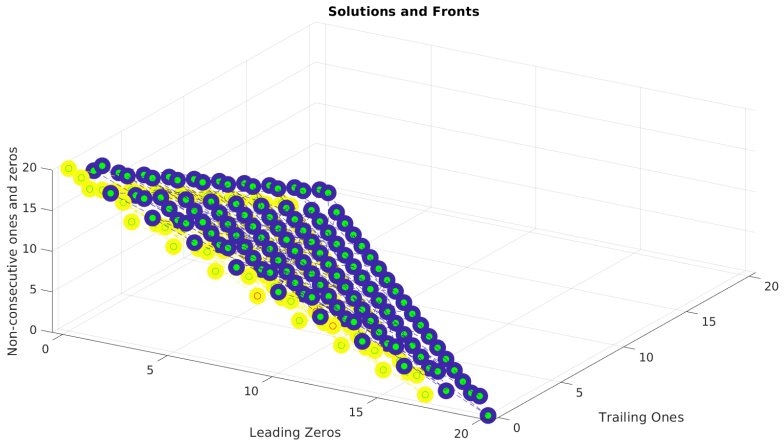

Figure 3: 3D plot for problem including the third objective.