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

YOUR NAME, `your.email@h-brs.de`

June 27, 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)

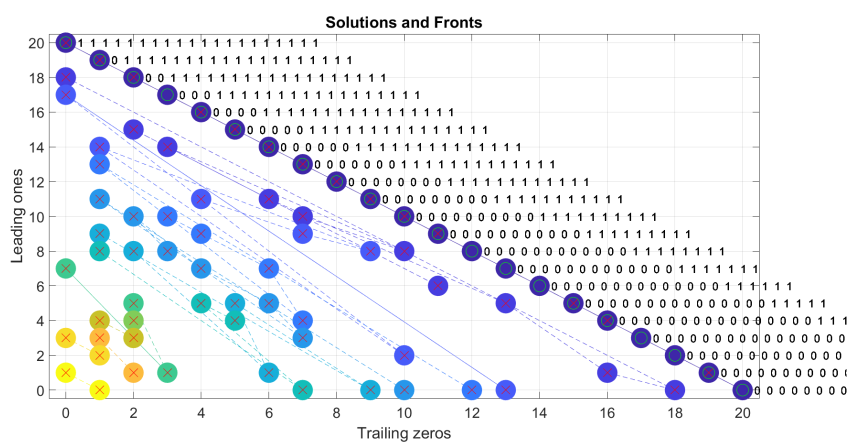
- (50pts) Implement NSGA-II to find all non-dominated solutions to the trailing ones, leading zeros problem.
  - Bitstring with length 20
  - Population size of 100
  - Generations 100
  - Hints:
    - Crossover and mutation can be performed just as in other bit string problems, e.g. one-max
    - The `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`)
- (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).

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. For the population size of 10:
     - Generation Size: 100
     - Population Size: 10
     - Total time taken is: 21.984 seconds
  2. For the population size of 100:
     - Generation Size: 100
     - Population Size: 100
     - Total time taken is: 53.8137 seconds
  3. For the population size of 1000:
     - Generation Size: 100
     - Population Size: 1000
     - Total time taken is: 2318.919 seconds or 38.64 mins

- (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?



plot for 100 generations and 100 population

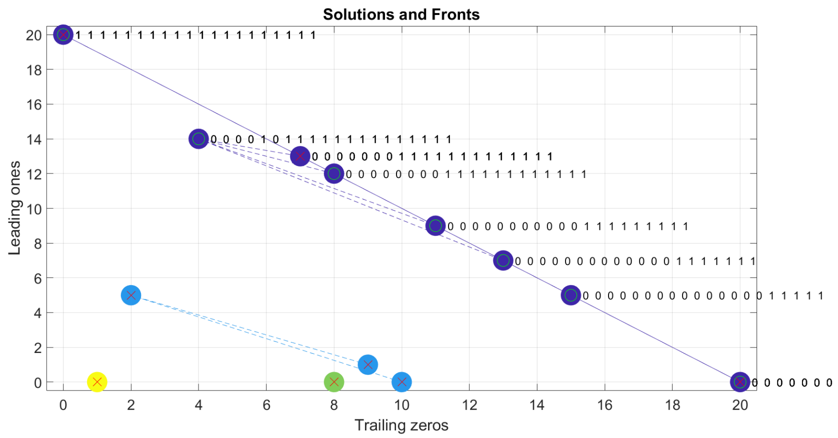


Figure 1: plot for 1000 generations and 10 population

From the above plots we can infer that it is better to have a larger population size as this gives us a more complete range of the solutions in the pareto front.

- (5pts) Imagine you were to replace the objective of "leading zeros" with "largest binary number". Predict the result, and give your reasoning.
  The final plot of the pareto front would then flip along the y-axis with the individual having the maximum fitness for both the leading zeros and largest binary number occupying the position 20,20 on the plot and the individual with the least fitness occupying the position 0,0 on the same plot.

- (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 population size and number of generations should be increased accordingly.

- (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?
  Because the computational complexity is $O(MN^2)$ in the worst case(given a number of fitness functions), thus greatly reducing the run time for these high-dimensional MOO problems.

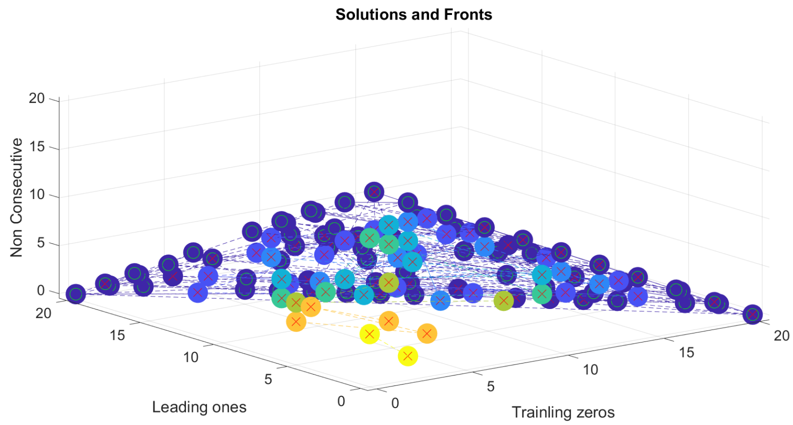


Figure 2: The plot obtained for non-consecutive zeros and ones with population and genarations set to 100.

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)
   A total of 199 possible non-dominated solutions can be possible.

2. What changes did you make to the algorithm and hyperparameters to get a good result?
   We changed the population size to 100 as any lower population size doesn't yield a complete set of solutions.We also added a separate fitness function to compute the fitness for non-consecutive ones and zeros.

3. List the solutions in your 1st front. Are they all Pareto optimal? How complete is your front (in percentage, based on #1)
   The list of solutions has been attached as a table at the end of the report.The front is 69 percent complete and 138 solutions are pareto optimal.

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

Table 1: List of the solutions

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