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 and MATLAB's quicksort built in `sortrows` function. How long does 100 generations take with each approach when using a population size of:

  – We take average of time taken over 10 experiments.

  1. 10:
     – Quicksort: 0.056817
     – Fast non-dominated sort: 0.082453
  2. 100:
     – Quicksort: 0.43277
     – Fast non-dominated sort: 1.6236
  3. 1000:
     – Quicksort: 4.9949
     – Fast non-dominated sort: 111.2765

- (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? Figure 1

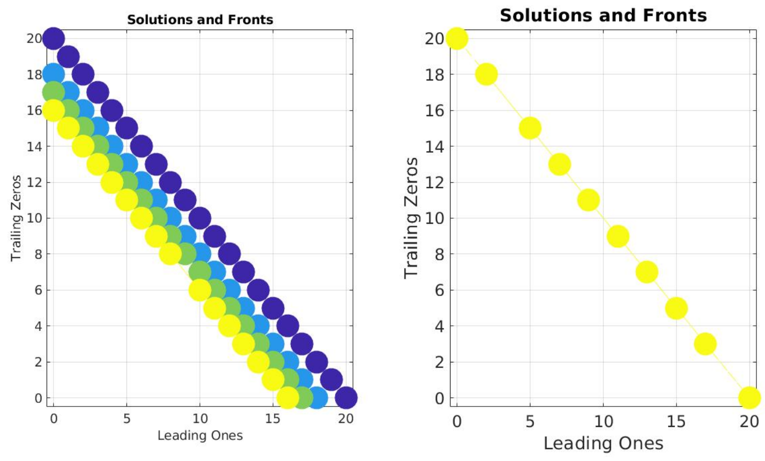

Figure 1: NSGA2 with 100 pop and 100 gen VS. NSGA2 with 10 pop and 1000 gen for solving leading ones trailing zeros problem with bitstring of length 20.

- (5pts) Imagine you were to replace the objective of "leading zeros" with "largest binary number". Predict the result, and give your reasoning.

- (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?
- (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?

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)
2. What changes did you make to the algorithm and hyperparameters to get a good result?
3. List the solutions in your 1st front. Are they all Pareto optimal? How complete is your front (in percentage, based on #1)
4. Plot the end result in 3D. (Use `plot3` or `scatter3`)

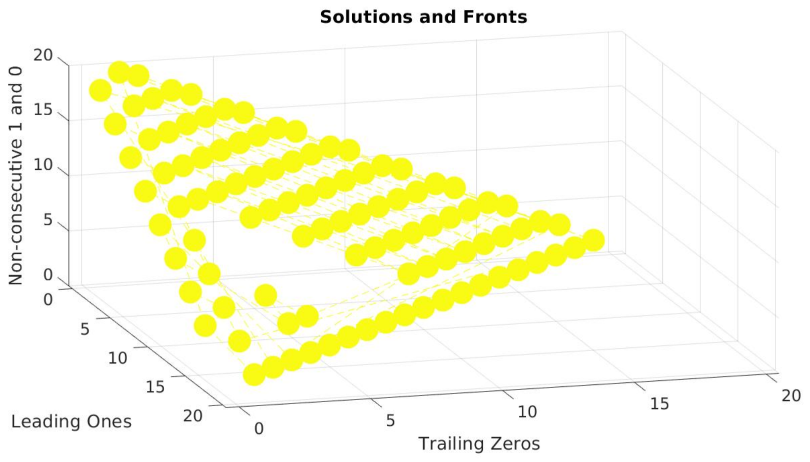

Figure 2: NSGA2 algorithm with 3 fitness (leading ones, trailing 0 and non-consecutive 0 and 1) ran for 100 gen with 100 pop for bitstring of size 20.