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

Djordje Vukcevic, djordje.vukcevic@smail.inf.h-brs.de
Supriya Vadiraj,supriya.vadiraj @smail.inf.h-brs.de

June 28, 2018

1 The Assignment

1.1 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).

1.2 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: 1.762199 seconds
  2. 100: 173.002463 seconds
  3. 1000: 16759.334529 seconds

- (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?

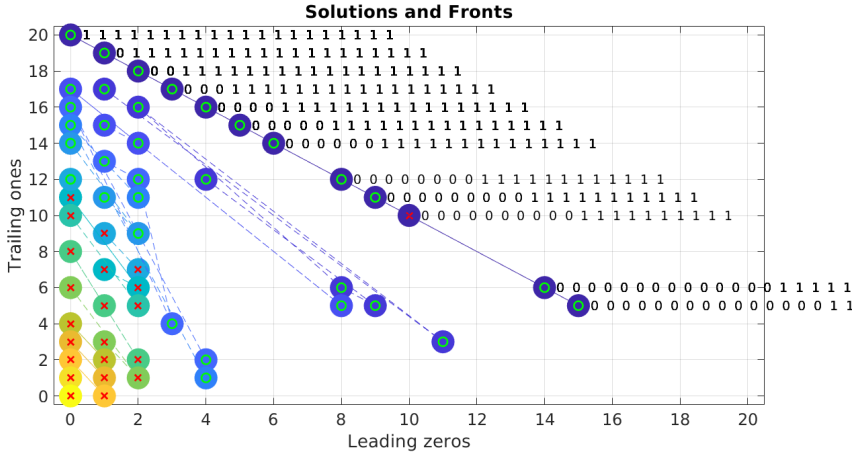

Figure 1: Population size = 100, while number of generations = 100

Answer: For the case when the algorithm is set with population size of 10 and number of generations 1000 (second scenario - figure 2), the algorithm finds optimal solutions in some generations, but not all found

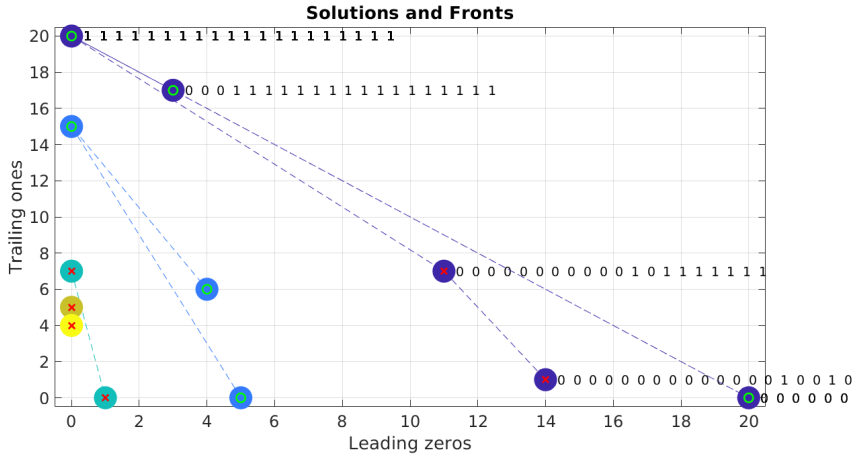

Figure 2: Population size = 10, while number of generations = 1000

optimal solutions are kept until the end of algorithm's run. This is due to fact that the space for exploration is very small compared to the another case. Here, some optimal solutions get "destroyed" over time, i.e. in this case, the algorithm performs much more of exploitation, rather than exploration. However, in the first scenario the population size is 100 (figure 1) and now the algorithm has the chance to explore bigger solution space and thus converge with more optimal solutions.

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

  Answer: With the "largest binary number" objective function, the algorithm would converge to the single solution, i.e. solution string that contains all ones. The reason for this is the fact that as we add more ones to the bit string, its value becomes bigger and this satisfies both objective functions.

- (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?

  Answer: If we were to add a third objective: "non-consecutive ones and zeros", the population size and number of generations should be increased in the hyperparameters 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?

  Answer: Non-dominated sorting consumes more runtime in NSGA-II

when there is a larger population size and a greater maximum generation are adopted . The computational cost will become much higher for larger population size or/and larger number of objectives, which in worst case computational complexity is $O(MN^2)$. Inorder to address this issue, a faster sort has been developed to improve the efficiency, thus reducing the runtime.

1.3 ** 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)

   Answer: We were able to generate 43 unique solutions for a popsize of 200.

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

   Answer: The main changes were increasing the population size, to a greater value of 200, as well as the fitness function was changed to compute non-consecutive zeros and ones.

3. List the solutions in your 1st front. Are they all Pareto optimal? How complete is your front (in percentage, based on #1) **Answer**: The solutions in the first front have been mentioned below

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

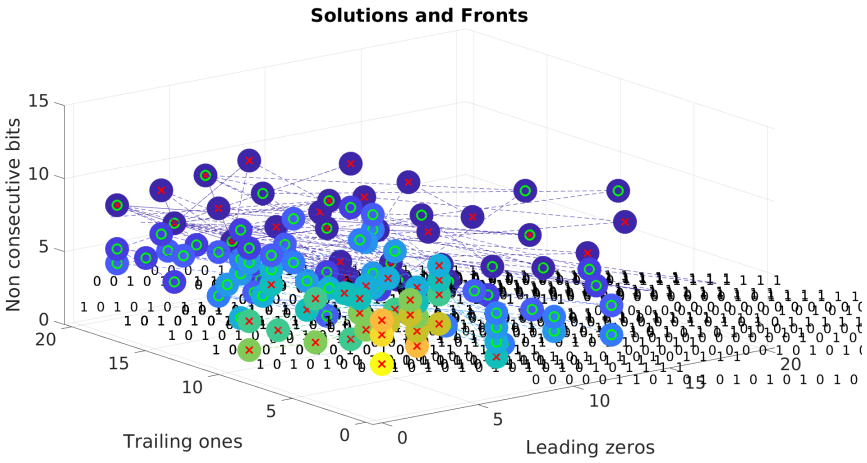

Table 1: My caption

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