Heuristic Project 2

Rugved Hattekar

April 2020

1 Introduction

The Goal of this assignment is to implement Multi Objective Genetic Algorithm (MOGA) using python. MOGA is a state of the art algorithm and is used extensively in the Optimization industry. Most of the multi objective algorithms can be solved as a single objective problem but it cannot be assumed that the goal of the multi objective algorithm is to find out the best optimal solution corresponding to the given objective function but rather the goal is to find the best solution by using trade-off between the solutions. In this assignement the MOGA is implemented using Python and nine Multi-Objective Functions listed below.

2 Pareto Optimality

Most multi-objective algorithms use the concept of dominance in their search. A state of affairs is Pareto-optimal (or Pareto-efficient) if and only if there is no alternative state that would make some people better off without making anyone worse off.

3 MOGA

Fonseca and Fleming first introdued a multi objective GA which used the non dominated classification of GA population. The MOGA only differs from a statndard GA in the way fitness is assigned to each solution in the population. The rest of the algorithm which is almost similar to the standard GA Algorithm and uses Stochastic universal selection, a double point crossover and Mutation. The MOGA uses Non-Dominated Sorting in the fitness function and there are number of steps to calculate the fitness function of MOGA.

3.1 Terminalogies used in MOGA Fitness Function

Niche Count:- In order to mintain the diversity amoung non-dominated solutions, the concept of niche count was introduced niching amoung solutions of each rank. It is nothing but the measure of densely populated the solution is.

Scaled Fitness: Dividing the assigned fitness value by niche count reduces the fitness of each solution, In order to keep the average fitness constant the scaled fitness is introduced.

3.2 Steps to calculate fitness function in MOGA

STEP 1: Calculate the objective function values from the randomaly generated values of the design variables.
STEP 2: Once the objective function values of more than 2 functions are available generate the rank of the functions using Non-Dominated Sorting.
STEP 3: Calculate the average fitness value by counting number of same rank solution and dividing each element by the count so every solution has a same fitness value.
STEP 4 Calculate the Niche Count of the solution by computing the normalized euclidean distance between two solutions. Multiply the value by sharing distance. The euclidean distance is calculated between the same rank solutions.The formula for euclidean distance is given below.

STEP 5 Calculate the shared fitness function using following formula. In the following formula μ stands for count of same rank solutions, N stands for total number of populations.

STEP 6 Calculate the scaling fitness function by scaling the fitness value equal to average fitness function using following equation. This can also be used as check the calculations. As average fitness should be equal to average fitness.

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2},$$

(a) Normalized Euclidean Dist.

$$F_i = N - \sum_{k=1}^{r_i - 1} \mu(k) - 0.5(\mu(r_i) - 1).$$

(b) Shared Fitness Formula

Figure 1: Formula to calculate fitness function

STEP 7 Return the scaling fitness value to the mating pool selector in our case it is Stochastic Roulette Wheel. And then further steps are as same as Binary GA as Two point crossover and Mutation.

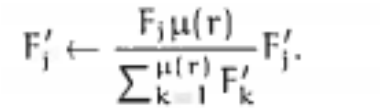


Figure 2: Scaling Formula

4 List of Problems Used in the Project

- 1. Zitzler–Deb–Thiele's function 1
- 2. Zitzler–Deb–Thiele's function 2
- 3. Zitzler–Deb–Thiele's function 3
- 4. Zitzler–Deb–Thiele's function 4
- 5. Zitzler–Deb–Thiele's function 5
- 6. Zitzler–Deb–Thiele's function 6
- 7. Schaffer function 1
- 8. Schaffer function 2
- 9. Disc Brake Design Problem
- 10. Binh and Korn Probelm

5 Results of Each Objective Function

In this section, let's look at the individual solutions of each objective function.

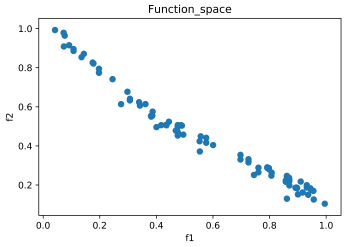
5.1 Zitzler–Deb–Thiele's function 1

ZDT- 1
$Number of Design Variables = 2$
$Range of Design Variables = [0, 1]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

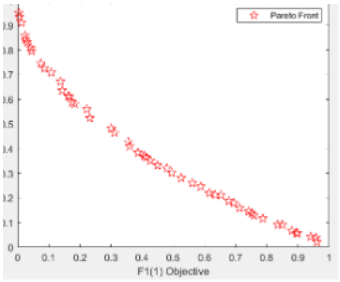
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} \end{cases}$$

Figure 3: ZDT-1 Function

Lets look at the pareto front of this function:



(a) Python ZDT-1



(b) Matlab ZDT-1 global optimization toolbox

Figure 4: ZDT Comparison

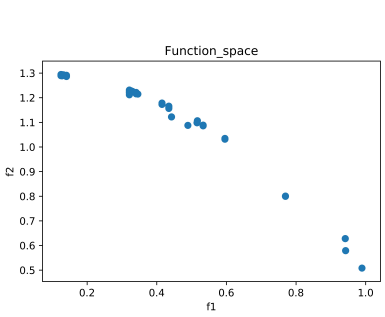
5.2 Zitzler–Deb–Thiele's function 2

ZDT- 2
$Number of Design Variables = 2$
$Range of Design Variables = [0, 1]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

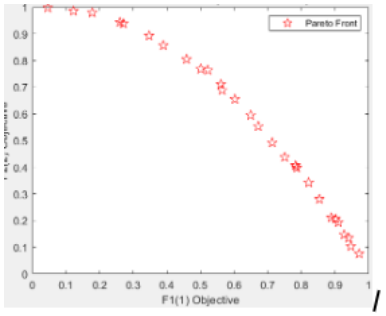
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2 \end{cases}$$

Figure 5: ZDT-1 Function

Lets look at the pareto front of this function:



(a) Python ZDT-2



(b) Matlab ZDT-2 global optimization toolbox

Figure 6: ZDT-2 Comparison

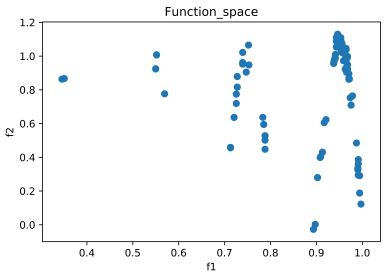
5.3 Zitzler–Deb–Thiele's function 3

ZDT- 3
$Number of Design Variables = 2$
$Range of Design Variables = [0, 1]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

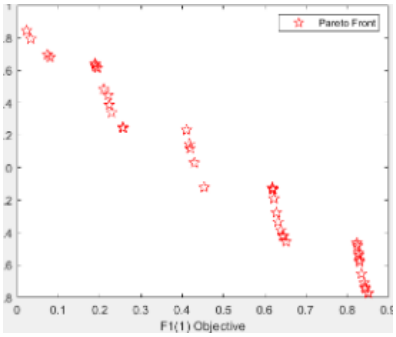
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right) \sin(10\pi f_1(\boldsymbol{x})) \end{cases}$$

Figure 7: ZDT-3 Function

Lets look at the pareto front of this function:



(a) Python ZDT-3



(b) Matlab ZDT-3 global optimization toolbox

Figure 8: ZDT-3 Comparison

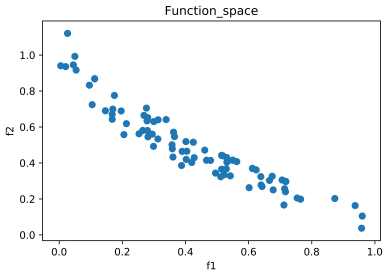
5.4 Zitzler–Deb–Thiele's function 4

ZDT- 4
$Number of Design Variables = 2$
$Range of Design Variables X1 = [0, 1], X2 = [-5, 5]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

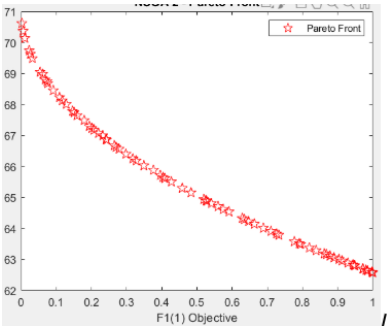
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) = 91 + \sum_{i=2}^{10} \left(x_i^2 - 10\cos(4\pi x_i)\right) \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} \end{cases}$$

Figure 9: ZDT-4 Function

Lets look at the pareto front of this function:



(a) Python ZDT-4



(b) Matlab ZDT-4 global optimization toolbox

Figure 10: ZDT-4 Comparison

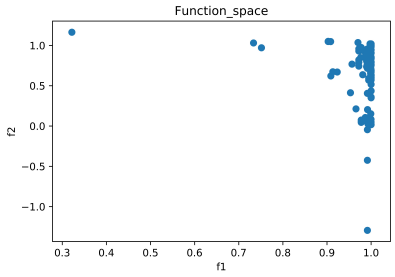
5.5 Zitzler–Deb–Thiele's function 6

ZDT- 6
$Number of Design Variables = 2$
$Range of Design Variables X1, X2 = [0, 1]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

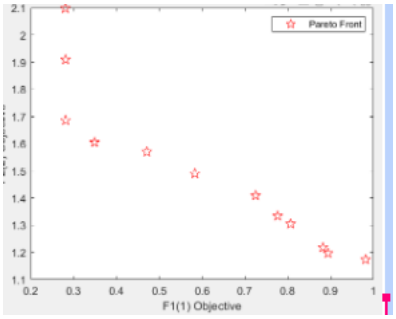
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) = 1 + 9\left[\frac{\sum_{i=2}^{10} x_i}{9}\right]^{0.25} \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2 \end{cases}$$

Figure 11: ZDT-6 Function

Lets look at the pareto front of this function:



(a) Python ZDT-6



(b) Matlab ZDT-6 global optimization toolbox

Figure 12: ZDT-6 Comparison

5.6 Zitzler–Deb–Thiele's function 5

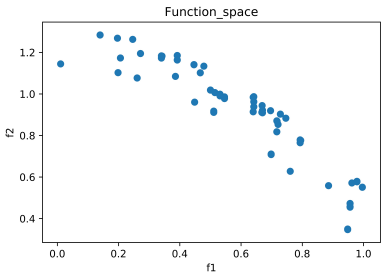
ZDT- 6

$Number of Design Variables = 2$
$Range of Design Variables X1, X2 = [0, 1]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

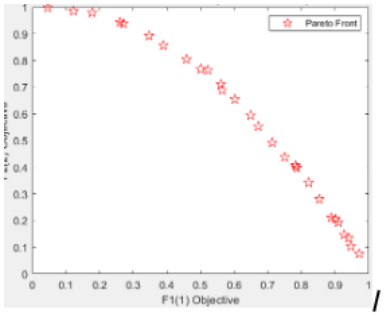
ZDT5	11	$x_i \in [0,1]$, 30 bit resolution $x_i \in [0,1]$, 5 bit resolution	$f_1 = 1 + u(x_1)$ $u(x_i)$ = the number of ones in the bit vector form of x_i $g = \sum_{i=2}^{m} v(u(x_i)),\quad h = \frac{1}{f_1}$ $v(u(x_i)) = \begin{cases} 2 + u(x_i) & \text{if } u(x_i) < 5 \\ 1 & \text{if } u(x_i) = 5 \end{cases}$ $f_2 = h.g$

Figure 13: ZDT-5 Function

Lets look at the pareto front of this function:



(a) Python ZDT-5



(b) Matlab ZDT-5 global optimization toolbox

Figure 14: ZDT-5 Comparison

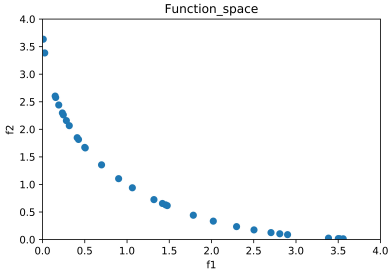
5.7 Schaffer function 1.

Schaffer function 1
$Number of Design Variables = 1$
$Range of Design Variables X1 = [-10, 10]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

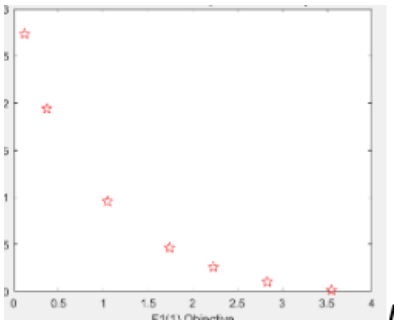
$$\text{Minimize} = \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$

Figure 15: Schaffer function 1 Function

Lets look at the pareto front of this function:



(a) Python Schaffer function 1



(b) Matlab Schaffer global optimization toolbox

Figure 16: Schaffer function Comparison

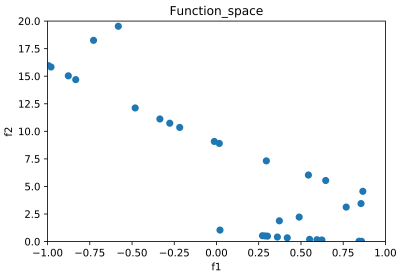
5.8 Schaffer function 2

Schaffer function 2
$Number of Design Variables = 1$
$Range of Design Variables X1 = [-5, 10]$
$Mutation Probability = 0.05$
$Cross - over probability = 0.85$
$Number of Runs = 10$

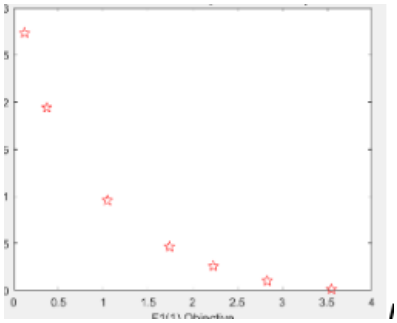
$$\text{Minimize} = \begin{cases} f_1(x) = \begin{cases} -x, & \text{if } x \leq 1 \\ x-2, & \text{if } 1 < x \leq 3 \\ 4-x, & \text{if } 3 < x \leq 4 \\ x-4, & \text{if } x > 4 \end{cases} \\ f_2(x) = (x-5)^2 \end{cases}$$

Figure 17: Schaffer function 1 Function

Lets look at the pareto front of this function:



(a) Python Schaffer function 1



(b) Matlab Schaffer global optimization toolbox

Figure 18: Schaffer function Comparison

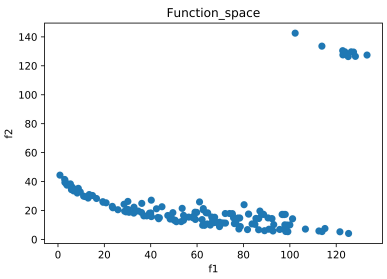
5.9 Bunh and Korn Function

Bunh and Korn Function
$Number of Design Variables = 2$
$Range of Design Variables X1 = [0, 5], X2 = [0, 3]$
$Mutation Probability = 0.10$
$Cross - over probability = 0.85$
$Number of Runs = 10$

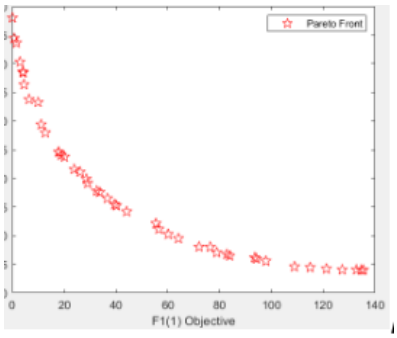
$$\text{Minimize} = \begin{cases} f_1(x, y) = 4x^2 + 4y^2 \\ f_2(x, y) = (x-5)^2 + (y-5)^2 \end{cases}$$

Figure 19: Bunh and Korn Function

Lets look at the pareto front of this function:



(a) Python Bunh and Korn Function



(b) Matlab Bunh and Korn Function global optimization toolbox

Figure 20: Bunh and Korn Function Comparison

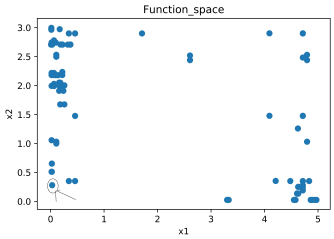


Figure 21: Bunh and Korn Variable Space

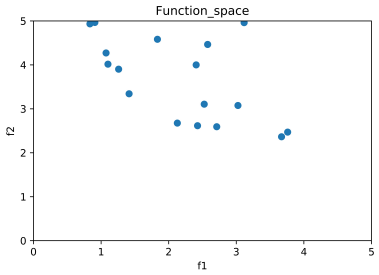
5.10 Disc Brake Problem

Disc Brake Problem
$Number of Design Variables = 4$
$Range of Design Variables X1 = [55, 80], X2 = [75, 110], X3 = [1000, 3000], X4 = [2, 20]$
$Mutation Probability = 0.010$
$Cross - over probability = 0.95$
$Number of Runs = 10$

$$\min f_1 = 4.9 \times 10^{-5}(x_2^2 - x_1^2)(x_4 - 1)$$
$$\min f_2 = \frac{9.82 \times 10^6(x_2^2 - x_1^2)}{x_3 x_4(x_2^3 - x_1^3)}$$

Figure 22: Disc Brake Problem

Lets look at the pareto front of this function:



(a) Python Disc Break Problem Function

6 Best Values at a glance

Let's Look at the best function values of each individual. This best value is choosen using min function in python after the best front is obtained by doing 10 runs of each multi objective function problem. Detail values of this front are attached in the respective folders of each function inside a text file. The Text file contains best fronts from which the min value given in the picture below can be deduced. The images we saw earlier in this report represents best front after 10 runs.

Function Name	Decision Variables	Function Best_value	Decision Variables	ss	Prob_c	Prob_m	Runs
ZDT 1	2	(0.01171875, 0.1417553150910899)	X1 = 0.1245, X2 = 0.1786354	0.5	0.85	0.05	10
ZDT 2	2	(0.0107421875, 0.346639303076)	X1 = 0.8876953125 X2 = 0.68423158	0.5	0.8	0.05	10
ZDT 3	2	(0.345703125, 0.02704959949513)	X1 = 0.2236328125 X2 = 0.08	0.5	0.8	0.05	10
ZDT 4	2	(0.6240234375, 0.444087713672673)	X1 = 0.044002558656 X2 = 0.3301110252	0.5	0.8	0.05	10
ZDT 5	2	(0.745703125, 0.0070495949513)	X1 = 0.231546 X2 = 0.09857251				
ZDT 6	2	(0.3433403359813558, 0.1223187751)	X1 = 0.9365234375 , X2 = 0.6658234	0.5	0.8	0.05	10
scaffer 1	1	(0.61334228515625, 1.5820922851)	X1 = -7.59765625	0.5	0.8	0.05	10
scaffer 2	1	(-0.90625, 0.00152587890625)	X1 = -4.921875	0.5	0.8	0.05	10
burhns and kurns	2	(1.292728424056, 4.44825745)	X1 = 0.3369140625 X2 = 2.0712890625	0.5	0.8	0.05	10
Disc Brake Problem	4	(2.042082077264786, 4.20477081595)	X1 = 55.512695 X2 = 75.5468 X3 = 1097.656 X4 = 16	0.5	0.8	0.05	10

Figure 24: Combined best results from all the problems

7 Conclusion

MOGA is applied to all the objective function given in the section 4. After appling the MOGA the importance of heuristic parameters such as sigma sharing, mutation probability as well as crossover probability and sigma share is understod. The MOGA is one of the powerful algorithms which can be applied to wide range of problem statements.

8 References

1 . K. Deb Muti-Objective Optimization using Evolutionery Algorithms.