2PM3 Assignment 2

Developing a Basic Genetic Optimization Algorithm in C

Pranav Chandrakumar

1 Introduction

In this document, I have reported the results of my Genetic Algorithm optimization for the specified parameters under section 2.1 and 2.2. There are two sets of reports, one with and the other without a stop parameter. Makefile compiling details are specified under section 2.3.

2 Programming Genetic Algorithm

2.1 Implementation with stop parameter

Table 1: Results with Crossover Rate = 0.5 and Mutation Rate = 0.05, Stop Parameter = 1e-16

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	0.8408951134611	,0.1290142653179	0.3188667823170	0.000138
100	100	0.0041885790434	-0.0216113566521	13.3124800832341	0.001837
1000	100	0.0021557556475	0.0015106704093	131.0616654315986	0.246868
10000	100	0.0001284666360	0.0007317145358	472.6003217156222	30.319508
1000	1000	0.0007025361995	0.0013058958581	235.1414489605118	1.245108
1000	10000	0.0000049383379	0.0000244309194	14181.3461519095908	32.391232
1000	100000	0.0000031758099	0.0000126962550	27011.4149945506505	134.987597
1000	1000000	0.0000004097819	0.0000004097819	409875.5385471744812	1095.037653

Table 2: Results with Crossover Rate = 0.5 and Mutation Rate = 0.2, Stop Parameter = 1e-16

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	0.0701843737951	0.2524453309608	0.5301841038	0.000155
100	100	0.0045661442003	0.0018759747044	68.4401959250	0.003849
1000	100	0.0012560631154	0.0009792391215	218.7082644045	0.239275
10000	100	0.0000464729964	0.0000079465098	7495.5523445043	33.875652
1000	1000	0.0002885446885	0.0003461819143	781.1984273771369	1.336697
1000	10000	0.0000064633786	0.0000307545997	11246.8641041767950	21.540129
1000	100000	0.0000011944212	0.0000028405338	114733.0776733889506	77.225209
1000	1000000	0.0000003934837	0.0000007310882	425834.9826973999151	689.806853

2.2 Implementation without stop parameter

Table 3: Results with Crossover Rate = 0.5 and Mutation Rate = 0.05, Stop Parameter = 0

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	0.1363142836542	0.0215624412622	1.2083097666982	0.000319
100	100	-0.0093076773962	0.0039229937847	31.9656856955569	0.006200
1000	100	0.0005886517467	0.0010334374387	293.9796163715323	0.327202
10000	100	0.0001881481149	0.0002336292528	1175.3077673575623	31.021747
1000	1000	0.0003827433103	0.0006501236933	465.3360619854337	3.372860
1000	10000	0.0000279094092	0.0000340212136	8031.1926229228711	33.300035
1000	100000	0.0000055739656	0.0000013224780	61712.8010209333442	328.095374
1000	1000000	0.0000000000000	0.0000002817250	1254955.5545887374527	2195.612149

Table 4: Results with Crossover Rate = 0.5 and Mutation Rate = 0.2, Stop Parameter = 0

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	0.0527940318234	0.0148404203424	4.2855297365215	0.000281
100	100	0.0035049696469	0.0078420643731	38.0822281753741	0.004251
1000	100	0.0000417861156	0.0002726772801	1278.3187190492444	0.243582
10000	100	0.0000255997292	0.0000796839595	4220.9732726185329	23.533426
1000	1000	0.0000119069591	0.0000022677704	29165.355134154910	2.410496
1000	10000	0.0000049732625	0.0000043911859	53287.2506191478605	24.133239
1000	100000	0.0000004819594	0.0000005145557	501476.9767742695403	241.672199
1000	1000000	0.0000001210719	0.0000000465661	2725545.7838061312213	2314.395854

As can clearly be seen from the difference in values between the tables in Section 2.2 and 2.1, when the `Stop Parameter` is set to 0, the algorithm is able to run through all generations. This ensures that the best possible solution is found for the given set of data. When `Stop Parameter` is set to a non-zero value, it may result in the program exiting prior to running through all generations, which highly detracts from the strength of the results. However, as can be seen from the reported values for CPU time, this premature exiting of the program also reduces the amount of time the program runs. This trade-off between result strength and program run-time is persistent throughout all optimization algorithms. From the results of the two sets of parameters, it appears as though setting `Stop Parameter` to 0 tends to ensure a stronger solution compared to setting a non-zero value.

2.3 Report and `Makefile` (3 points)

Here is the `Makefile` for this assignment:

```
CC = gcc
CFLAGS = -Wall -Wextra -O3
LDFLAGS = -lm
SOURCES = functions.h OF.c functions.c GA.c
OBJECTS = $(SOURCES:.c =.o)
EXECUTABLE = GA

all : $(EXECUTABLE)

$(EXECUTABLE):$(OBJECTS)
    $(CC) $(OBJECTS) -o $(EXECUTABLE) $(LDFLAGS)

%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@
clean :
    rm -f $(EXECUTABLE)
```

Standard `Makefile` format for this program. Run `make` in the terminal to create the executable file `GA`. `-lm` flag is included due to usage of functions from the library `<math.h>` in the `OF.c` and `GA.c` files. `-O3` is included as it's a flag which inherently optimizes the executable file while compiling (although it slightly increases compilation time). Source files are `functions.h`, `OF.c`, `functions.c`, and `GA.c`. Running `make clean` in the terminal will only remove the executable file, not the source code files as well.