¡Mechatronics 2MP3 Assignment 2¿

Developing a Basic Genetic Optimization Algorithm in C

¡Marcus De Maria¿

1 Introduction

Table 1: Results with Crossover Rate = 0.5 and Mutation Rate = 0.05

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	-0.013244	0.085305	0.431138	0.000041
100	100	-0.2786664	-0.078655	2.100787	0.002362
1000	100	-0.004097	0.098899	0.520986	0.022725
10000	100	-0.046203	-0.010222	0.192417	0.225258
1000	1000	-0.014522	-0.022853	0.096007	0.243871
1000	10000	-0.002328	-0.002969	0.011049	2.258876
1000	100000	0.000671	0.000951	0.003329	24.071972
1000	1000000	0.000025	0.000096	0.000096	267.165445

Table 2: Results with Crossover Rate = 0.5 and Mutation Rate = 0.2

Pop Size	Max Gen	Best Solution			CPU time (Sec)
		x_1	x_2	Fitness	
10	100	0.059030	0.065990	0.448721	0.000400
100	100	-0.024179	-0.025201	0.131000	0.002626
1000	100	0.020239	-0.074469	0.369694	0.025303
10000	100	-0.008048	0.013263	0.050277	0.250076
1000	1000	-0.005404	-0.006352	0.025440	0.246751
1000	10000	0.000733	0.000510	0.002546	2.523743
1000	100000	-0.000876	0.000205	0.002566	25.144203
1000	1000000	-0.000246	0.000180	0.000812	254.309610

1.1 Report and `Makefile` (3 points)

Open up VSCode (or use Ubuntu and type "code ." to access WSL VSCode). Use "cd" to get to the correct directory in the terminal for "Assignment2". Compile the code using ./GA 1000 10000 0.5 0.1 1e-16, or by directly using the makefile by typing make in the terminal. A build automation tool called a makefile is used to control the process of compiling source code files and linking them into executable applications. They outline the rules for constructing the final executable in addition to the dependencies between various source and header files. The make utility runs

the required compiler commands after reading the makefile to identify which parts of the code need to be recompiled. A makefile typically comprises of dependencies that indicate the relationships between source files and the finished result and rules that specify how the executable should be built. Typically, a rule consists of commands, dependencies, and a target. The file or action to be carried out is the target; files that the target depends on are called dependencies; and shell instructions are called commands used to build the target. The compiler (gcc) is represented by the CC variable in the example makefile that is provided; compiler flags like -Wall and -Wextra are included in CFLAGS; source files are listed in SOURCES; object files to be generated are specified in OBJECTS; the name of the final program is EXECUTABLE; and the math library (-lm) is included in LIBS. The executable is dependent upon the object files, which are dependent of the entire target. The created files are removed by the clean target, and the rule for object files describes how to compile source files into objects. The make command in the terminal can be used to build the program; it will compile the source code and create the executable automatically. Only files that have changed since the last build are recompiled by the make tool, which also examines dependencies and helps in efficient code development.

1.2 Improving the Performance - Bonus (+1 points)

1.3 Bonus (+3 points) - Only the fastest program!