Central Washington University, Department of Computer Science

CS471 Project3

Jason Stearns

May 8, 2018

1 Introduction

The purpose of this lab was to code two evolutionary algorithms, one bing the Genetic Algorithm and the other being the Differential Evolutionary Algorithm. The genetic algorithm uses a two dimensional array known as a population, the population consists arrays that represent each member of the population. Each of these arrays contains genes which are used by the algorithm to emulate genetics. In order to emulate the combining of genes there is a crossover function. The crossover function takes two parent vectors, a crossover rate and a randomly generated number between 0 and 1. This number is compared to the crossover rate. if the crossover rate is larger than the randomly generated number then the crossover is performed. Otherwise the child is a direct clone of the parent. after the crossover mutation is performed which ensures that the population doesnt stagnate by adding in noise. This results in two new child vectors which then replace their parents in the population. The process then repeats for all members of the population. Then once a new population has been generated the reduce function is called. This function takes the best N solution vectors from the new population and mixes them with the NS - N solutions from the old population resulting in an improved population.
The second algorithm (differential evolution) uses similar ideas from the previous algorithm but also improves upon it by using ten strategies for mutation and two different types of crossover functions. Differential evolution starts off the same way as the genetic algorithm, it generates a random population however this time the mutation is done between a parent vector and a noisy vector. The noisy vector is generated 10 different ways. For the purposes of experimentation in this lab, Strategies 1, 7 and 9 were used. Strategy 1 is best/1/exp. which means that the current best solution vector in the population is combined using a scalar factor with 2 other randomly selected solution vectors in the population and then it is combined

using exponential crossover. Exponential crossover is where a single crossover point is generated and the two parents genes are split along that point. Strategy 7 is Rand/1/bin is similar to strategy 1 however instead of the best solution vector being used, instead another random solution vector is used, then crossover is performed using binomial crossover, which is where crossover is randomly determined at every gene. Strategy 9 is best/2/bin, the difference being that now 4 random solution vectors are chosen to combine with the best solution vector.

2 RESULTS

2.0.1 DIFFERENTIAL EVOLUTIONARY ALGORITHM

The results of strategy 1 and strategy 2 are shown in tabel 3.1 as shown. It is clear that the minimum of function F3 is not optimal, since the range between the solutions is almost 10 million, when compared to iterative local search it is clear that iterative local search is a much better way of calculating minimums in fewer iterations. The evolutionary algorithm could improve further with more iterations. Up examining the average of both strategies it can also be shown that strategy 1 is better on average than strategy 7 when it comes to finding lower minimums. Upon further analysis of the functions themselves a pattern emerges where strategy 7 fairs better when used for functions that have larger maximums and smaller minimums while in the case of strategy 1, strategy 1 performed better when the range of the fitness function is smaller.After graphing the data for strategy 1 it became apparent that it stagnated to the point where it was giving same fitness for multiple returns. This stagnation was likely caused by suboptimal configuration of variables for mutation rate and crossover rate for this strategy.

2.0.2 GENETIC ALGORITHM

The results of the genetic algorithm were much more consistant than the Differential Evolutionary Algorithm. There was no stagnation however it seems like the Dejong and Rastrigin functions do not settle near their minimums found by local search, this could be due to suboptimal configuration of variables but upon closer examination of the functions rastrigin function coded for this experiment has two components transposed, making it a maximization function and not a minimization function, resulting in higher values than normal. This explains why local search found an optimal solution since it is not based on random improvements.

3 CONCLUSION

The data has shown that Evolutionary algorithms need many iterations in order to find optimal solutions, or they need to mutate and crossover frequently in order to maximize their effect. Also the Differential evolution strategies that us the best solution tend to improve faster since they are improving the best solution all the time instead which cuts down on randomness that pushes the search to a non-optimal solution. Another problem with Evolutionary algorithms is that they can stagnate which can lead to inaccurate results even for differential evolution.

Table 3.1: Computation comparison of DE1 and DE7

Problem	DE1					DE7				
	Avg	Median	Range	SD	T(s)	Avg	Median	Range	SD	T(s)
f_1	-221.5771333	-201.843	99.498	394.9	34.839	-1700.847667	-1726.795	1734.07	403.2750322	34.839
f_2	43799.416	43776	126.10292	702.5	22.73	18668.184	19734.45	15237.48	4160.821239	22.73
f_3	9959601333	9957700000	9974937.839	55600000	39.562	4422943667	4643215000	5430020000	1708103852	
f_4	592142.8	589830	5566.1582	22877	36.216	65955.45667	64004.8	70814.2	16378.52288	36.216
f_5	576.34346	576.244	0.4333	2.395	41.014	123.9977	128.2855	102.3355	27.56284582	41.014
f_6	-9.8581	-9.86145	0.100	0.63214	64.193	-18.03181	-18.052	2.2045	0.6095	64.193
f_7	163.49876	163.387	0.220370247	0.896	71.195	90.98624	91.45425	22.0665	5.189186085	71.195
f_8	43.0071	43.0071	1.42109E-14	0	52.96	195.2727667	194.662	102.71	25.01321223	52.96
f_9	36.1618	36.1618	1.42109E-14	0	79.326	272.3956333	275.1715	59.728	14.87481426	79.326
f_{10}	4175.094	4148	178.8716588	857.3	48.734	-2282.400667	-2312.205	1875.14	422.8699639	48.734
f_{11}	-5290.6156	-5297.43	94.76758	385.87	77.86	-1542.194667	-1486.04	1025.94	250.0891943	77.86
f_{12}	18.01111	18.0456	0.120943236	0.4965	49.686	8.606343333	8.60607	0.86041	0.189029583	49.686
f_{13}	-4.381197667	-4.56817	0.338916136	0.80131	47.728	-4.667221667	-4.63558	1.93491	0.409039992	47.728
f_{14}	-8.21516	-8.21516	0	0	47.823	-6.672137667	-6.45521	5.74824	1.460332783	47.823
f_{15}	-0.023969343	-0.0226396	0.001756523	0.0043763	78.951	-0.04044	-0.03939725	0.0341569	0.006946582	78.951

[1] , 3.4GHz Intel core i5-3570K, 8 GB RAM

Table 3.2: Computation of GA

Problem	GA				
	Avg	Median	Range	SD	T(s)
f_1	-1.15E+06	-1.13E+06	114635.2531	5.14E+05	34.553
f_2	6.37E+04	5.96E+04	18009.89774	8.33E+04	22.765
f_3	5.43E+10	3.18E+10	64564949312	2.47E+11	39.381
f_4	2.14E+05	1.96E+05	61068.76352	2.19E+05	36.236
f_5	4.02E+02	3.57E+02	112.6178775	5.11E+02	40.965
f_6	-1.53E+01	-1.51E+01	0.658330545	2.88E+00	63.962
f_7	1.26E+02	1.26E+02	11.83085504	4.80E+01	70.951
f_8	3.53E+02	3.47E+02	35.9757453	1.58E+02	52.824
f_9	3.37E+02	3.41E+02	11.94144434	4.54E+01	79.049
f_{10}	-1.49E+06	-1.48E+06	178865.4188	9.66E+05	48.766
f_{11}	-1.07E+06	-1.05E+06	136020.2282	5.11E+05	77.699
f_{12}	8.73E+00	8.69E+00	0.193480095	7.16E-01	49.35
f_{13}	-4.92E+00	-4.74E+00	0.640204819	2.86E+00	50.528
f_{14}	-1.10E+01	-1.09E+01	0.936520126	3.28E+00	47.697
f_{15}	-3.46E-02	-3.25E-02	0.006878137	3.89E-02	78.517

[1] , 3.4GHz Intel core i5-3570K, 8 GB RAM