CS 471 - Project 3

Andrew Dunn
Instructor: Dr. Donald Davendra
Central Washington University
Ellensburg, Washington

May 3, 2019

Abstract

In this document we will first introduce you to the problems that were tested and our test methodology. We will then display the results we gathered, and explain our conclusions based on the data analysis.

1 Introduction

For this experiment we tested 18 different mathematical functions that are n-dimension scalable. More information about all 18 functions can be found in Table 1. The program developed to test these functions can be configured with several different parameters, such as specifying the population and dimension sizes, search algorithm, and number of test iterations. In our tests we ran multiple experiments utilizing two different search methods in various dimension sizes. The latest version of our program utilizes multi-threading to decrease execution times. The tests we ran used 8 different threads, and all functions computed values using 64-bit floating point data types.

The testing hardware we used for this experiment is a desktop computer running Ubuntu 19.10 with an Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz and 32 GB of DDR3 @ 1600 MHz. Our testing program was written in C++ and compiled with CMake version 3.12.1 and g++ version 8.2.0. We are using the C++11 standard libraries with the built in Mersenne Twister random number generator to generate the population data between the given ranges for each function in Table 1.

Table 1: Experiment functions

f	Name	f(x*)	Dimensions	Range
1	Schwefel	0	$10, 20, 30$	$[-512, 512]^n$
2	De Jong 1	0	$10, 20, 30$	$[-100, 100]^n$
3	Rosenbrocks Saddle	0	$10, 20, 30$	$[-100, 100]^n$
4	Rastrigin	0	$10, 20, 30$	$[-30, 30]^n$
5	Griewangk	0	$10, 20, 30$	$[-500, 500]^n$
6	Sine Envelope Sine Wave	$-1.4915(n-1)$	$10, 20, 30$	$[-30, 30]^n$
7	Stretch V Sine Wave	0	$10, 20, 30$	$[-30, 30]^n$
8	Ackley One	$-7.54276 - 2.91867(n-3)$	$10, 20, 30$	$[-32, 32]^n$
9	Ackley Two	0	$10, 20, 30$	$[-32, 32]^n$
10	Egg Holder	$-$	$10, 20, 30$	$[-500, 500]^n$
11	Rana	$-$	$10, 20, 30$	$[-500, 500]^n$
12	Pathological	$-$	$10, 20, 30$	$[-100, 100]^n$
13	Michalewicz	$0.966n$	$10, 20, 30$	$[0, \pi]^n$
14	Masters Cosine Wave	$1-n$	$10, 20, 30$	$[-30, 30]^n$
15	Quartic	0	$10, 20, 30$	$[-100, 100]^n$
16	Levy	0	$10, 20, 30$	$[-10, 10]^n$
17	Step	0	$10, 20, 30$	$[-100, 100]^n$
18	Alpine	0	$10, 20, 30$	$[-100, 100]^n$

2 Methods

Our method was to test two different evolutionary algorithms, genetic algorithm and differential evolution. Both were executed 50 times, and the results of each of the 50

experiments were recorded. We used population sizes of 200 vectors in 30 dimensions. The population vectors were generated using a uniformly stochastic random number generator. Each function population is generated between certain bounds as seen in Table 1. Our goal is to produce a fitness value from a generated population that is close as possible to the optimal f(x*) for each function. All fitness data is recorded and ran through a few different statistical analysis functions: average, standard deviation, range, and median. We also recorded the total amount of time in milliseconds that it takes to run each iteration of the evolutionary algorithms. Finally, the genetic algorithm was executed using one selection method, roulette wheel, while the differential evolution algorithm was executed using 10 different strategies.

3 Execution Times

Table 2: Execution Times (ms) - Genetic Algorithm - Single Experiment

f(x)	Min	Max
1	39.84	79.52
2	28.52	42.82
3	27.75	47.31
4	35.62	57.71
5	42.60	81.04
6	36.38	58.92
7	70.19	103.36
8	53.31	72.98
9	81.80	126.99
10	45.37	67.95
11	75.64	98.82
12	43.31	62.58
13	64.38	96.70
14	55.38	122.03
15	27.12	62.50
16	39.53	61.62
17	30.38	43.88
18	38.20	57.85

Table 3: Execution Times (ms) - Differential Evolution - Single Experiment

	Strategy 1		Strategy 2		Strategy 3		Strategy 4	
f(x)	Min	Max	Min	Max	Min	Max	Min	Max
1	24.93	46.95	23.43	59.13	24.14	72.95	29.29	61.75
2	8.13	24.47	7.89	21.30	7.70	26.95	11.96	36.05
3	7.85	35.34	10.31	24.25	9.46	23.97	13.31	28.12
4	24.54	45.39	24.12	46.03	21.54	39.80	27.57	51.77
5	27.04	45.31	28.54	48.73	27.94	49.47	29.89	64.43
6	22.96	46.14	25.20	41.45	20.69	40.93	33.04	53.59
7	55.47	91.13	58.46	94.24	55.51	90.03	59.52	103.83
8	27.70	56.76	35.57	61.58	28.43	75.44	39.71	63.63
9	69.42	114.24	74.51	114.07	65.37	116.28	70.42	130.25
10	37.54	68.52	42.79	63.86	40.47	83.99	42.49	73.16
11	67.54	108.57	67.52	103.95	71.91	99.28	77.05	111.02
12	22.17	52.48	27.72	47.94	23.36	47.97	27.89	49.26
13	50.32	85.17	52.65	87.36	54.23	86.97	52.83	98.50
14	38.71	74.71	39.14	75.89	42.15	72.95	43.59	80.53
15	8.37	19.15	8.02	24.02	8.39	23.10	10.83	32.54
16	20.92	45.73	21.98	46.24	23.21	43.89	32.71	54.12
17	7.43	27.15	9.02	23.69	7.41	21.70	11.32	33.19
18	21.36	45.54	22.57	49.20	19.94	45.90	28.07	73.51

Table 4: Execution Times (ms) - Differential Evolution - Single Experiment

	Strategy 5		Strategy 6		Strategy 7		Strategy 8	
f(x)	Min	Max	Min	Max	Min	Max	Min	Max
1	30.20	51.86	34.08	76.79	35.57	77.74	35.66	88.04
2	14.30	31.22	16.46	47.41	18.78	44.67	15.33	46.44
3	13.70	33.13	16.25	37.55	20.98	48.33	16.79	38.46
4	29.40	73.35	27.05	58.60	35.11	90.37	28.26	55.34
5	35.44	61.63	30.07	60.16	34.30	75.18	31.11	60.58
6	32.59	48.24	30.05	61.64	37.85	77.53	28.11	61.55
7	66.97	125.03	75.13	122.98	64.71	136.65	71.49	131.13
8	38.37	93.31	48.32	105.08	53.84	125.89	35.08	81.14
9	68.85	140.30	76.68	120.98	80.49	177.30	82.63	120.30
10	41.27	72.83	45.56	103.10	56.64	130.36	46.40	84.78
11	76.53	149.60	81.73	188.05	82.55	129.30	77.47	148.73
12	29.15	75.77	33.61	68.71	38.20	78.07	34.80	62.52
13	56.94	134.15	71.53	161.54	69.38	117.66	61.31	106.52
14	43.08	78.05	52.55	102.47	60.73	121.36	51.14	85.58
15	10.86	31.90	16.42	39.94	17.83	36.28	18.25	39.31
16	32.01	68.75	28.73	69.78	33.30	76.85	29.83	53.32
17	13.05	33.51	20.03	50.28	16.51	42.26	19.34	37.22
18	29.99	78.17	34.65	67.53	32.06	83.19	30.62	63.66

Table 5: Execution Times (ms) - Differential Evolution - Single Experiment

	Strategy 9		Strategy 10	
f(x)	**Min**	**Max**	**Min**	**Max**
1	34.80	76.68	40.71	65.14
2	19.59	46.80	23.98	46.61
3	19.78	61.00	21.13	52.75
4	35.59	63.24	32.56	68.91
5	36.85	66.32	38.58	72.18
6	32.49	65.16	40.88	66.08
7	67.32	117.45	76.90	118.36
8	48.00	79.23	51.86	82.90
9	76.46	127.71	86.39	132.34
10	49.19	81.94	52.29	84.82
11	84.25	116.88	80.71	121.03
12	34.16	64.33	35.77	74.88
13	70.87	121.56	59.83	103.99
14	50.64	91.17	58.20	91.28
15	24.15	47.88	19.48	48.57
16	33.41	89.86	37.46	69.67
17	24.80	44.46	20.89	43.23
18	31.83	78.19	33.91	80.51

The first thing we are going to look at is the execution times in milliseconds for our tests. Each min and max value in the execution time tables is for a single experiment of the algorithm. First, take a look at Table 2 containing the execution times of the genetic algorithm. When you compare these times to differential algorithm in Table 3, Table 4, and Table 5, you'll notice that the minimum times tend to be on average greater than the minimum execution times for differential evolution for the same function. Overall however all times are fairly consistently and under 200 milliseconds. Recall from our last experiments with local search, where some functions had execution times ranging from less than a millisecond to over 40 seconds. Genetic algorithm and differential evolution perform much better from a execution time standpoint and have much less variance between their min and max times.

This is most likely due to stopping criteria of these algorithms versus local search. Local search would keep running until the population is no longer improving, which for some of the functions could take a very long time. In contrast genetic algorithm and differential evolution run for a fixed number of generations, which in the end created similar execution times for all functions.

4 Project 2 Results Review

To recall the results from project 2, you can find the them in this section.

In this section you will find all results and statistical analysis for both the blind search and local search algorithms in 30 dimensions. Note that the statistics for each function f_n is over the course of 30 iterations. We did not differentiate between local search ($1\ iteration$) and iterative local search ($> 1\ iterations$) because single iteration local search is just a subset of iterative local search. Table entries colored in red are column maximums, green entries are those that are closest to zero for that column, and blue entries highlight functions that produce the best results, i.e. closest to their optimal value f(x*) for that function.

Table 6: Blind Search - 30 Dimensions - 30 Iterations

f	Average	Std. Deviation	Range	Median	Avg. Time (ms)
1	10502.84	383.22	1592.69	10465.35	0.08
2	68757.26	7670.73	33798.20	68587.80	0.03
3	3.17E+10	5.59E+09	2.46E+10	3.09E+10	0.03
4	1758367.33	250518.28	1065510.00	1768355.00	0.06
5	425.79	44.73	209.79	432.34	0.07
6	-24.05	0.90	3.83	-23.90	0.06
7	76.07	3.76	14.00	76.78	0.13
8	460.59	32.39	139.24	467.33	0.08
9	541.21	12.85	62.85	543.50	0.18
10	-3449.60	793.66	3781.37	-3229.55	0.11
11	-2231.85	569.79	2746.48	-2228.94	0.18
12	13.66	0.21	0.77	13.66	0.06
13	-6.13	0.70	3.26	-5.95	0.12
14	-0.57	0.33	1.67	-0.62	0.09
15	4.32E+09	8.37E+08	4.11E+09	4.42E+09	0.03
16	258.74	37.28	184.23	257.53	0.06
17	69049.46	8105.32	36845.40	70051.20	0.03
18	675.36	60.63	316.87	673.45	0.05

Table 7: Local Search - 30 Dimensions - 30 Iterations

f	Average	Std. Deviation	Range	Median	Avg. Time (ms)
1	5134.00	521.29	2144.33	5011.16	108.88
2	0.09	0.00	0.01	0.09	0.74
3	3.15E+10	3.90E+09	1.62E+10	3.17E+10	0.03
4	1903896.00	227903.55	880340.00	1960160.00	0.08
5	1.00	0.00	0.00	1.00	48424.05
6	-23.93	0.74	3.00	-23.67	0.08
7	34.58	4.26	20.06	33.13	607.07
8	352.28	24.91	108.66	352.02	3.97
9	498.68	15.16	63.40	500.62	14.34
10	-11490.52	1844.84	9890.88	-11958.25	644.98
11	-6731.40	1768.29	7128.38	-7097.64	1773.02
12	13.66	0.26	1.04	13.72	0.09
13	-5.99	0.68	3.01	-5.93	0.19
14	-0.87	0.37	1.35	-0.74	1206.63
15	4.38E+09	1.12E+09	4.52E+09	4.28E+09	0.03
16	51.85	17.96	55.98	41.92	68.71
17	9.20	0.06	0.32	9.21	0.25
18	514.43	105.54	402.18	502.35	0.12

5 New Results

In this section you will find all of our new experiment results from the genetic algorithm and differential evolutionary algorithm. For each table we have also provided an example line graph showing the population improvement over time for the Schwefel function, function 1 over all 100 generations.

Figure 1: Example of Population Improvement

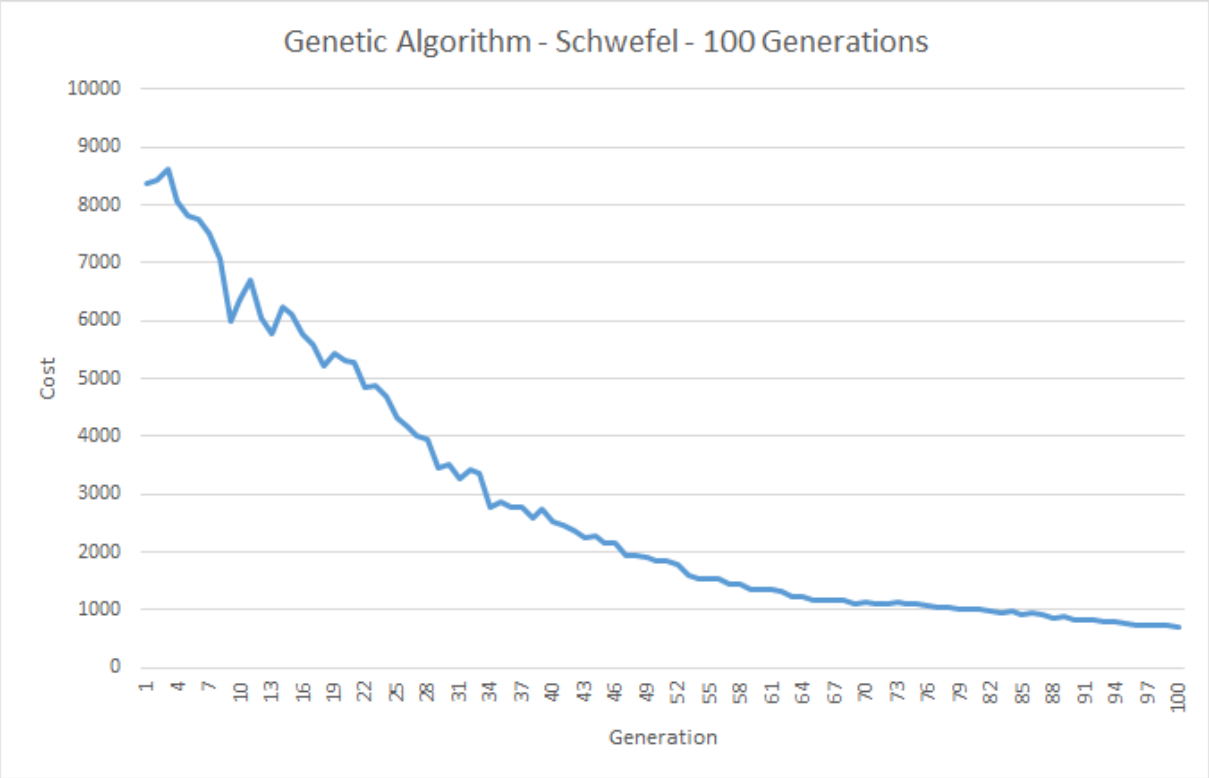

Table 8: Genetic Algorithm - Roulette Wheel - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	1409.284653	358.375724	1451.37079	1591.5312	699.11931
2	857.807546	308.67808	1131.92014	512.86124	325.79366
3	40871057.69	34468711.68	181464667.1	34060619.5	5899992.9
4	-19799.06439	14723.4247	73989.943	-25740.2785	-41248.367
5	6.166095	2.486216	10.55429	5.16473	2.73055
6	-38.566841	0.863623	4.021618	-38.565934	-40.597703
7	31.309248	0.647397	3.640922	31.226281	30.018
8	-37.67213	11.877742	44.540614	-42.256097	-52.303887
9	117.773244	13.508431	49.593823	120.221655	97.118497
10	-12407.26667	1249.221689	6005.2914	-12333.5495	-15402.188
11	-7882.019022	598.526	3010.474	-7846.7365	-9722.9117
12	6.946521	0.466502	1.885526	6.965459	6.022934
13	-24.842781	1.011068	5.231098	-25.195884	-26.286702
14	-9.715474	1.867635	8.888868	-7.1306	-14.446877
15	6859130.134	5289334.142	28712241.8	4203369.7	1543819.2
16	5.771988	1.962127	8.886648	5.315028	1.971973
17	859.679765	353.307661	1470.88086	679.09807	335.08444
18	35.976585	8.599594	36.352284	35.721819	17.434286

Figure 2: Example of Population Improvement

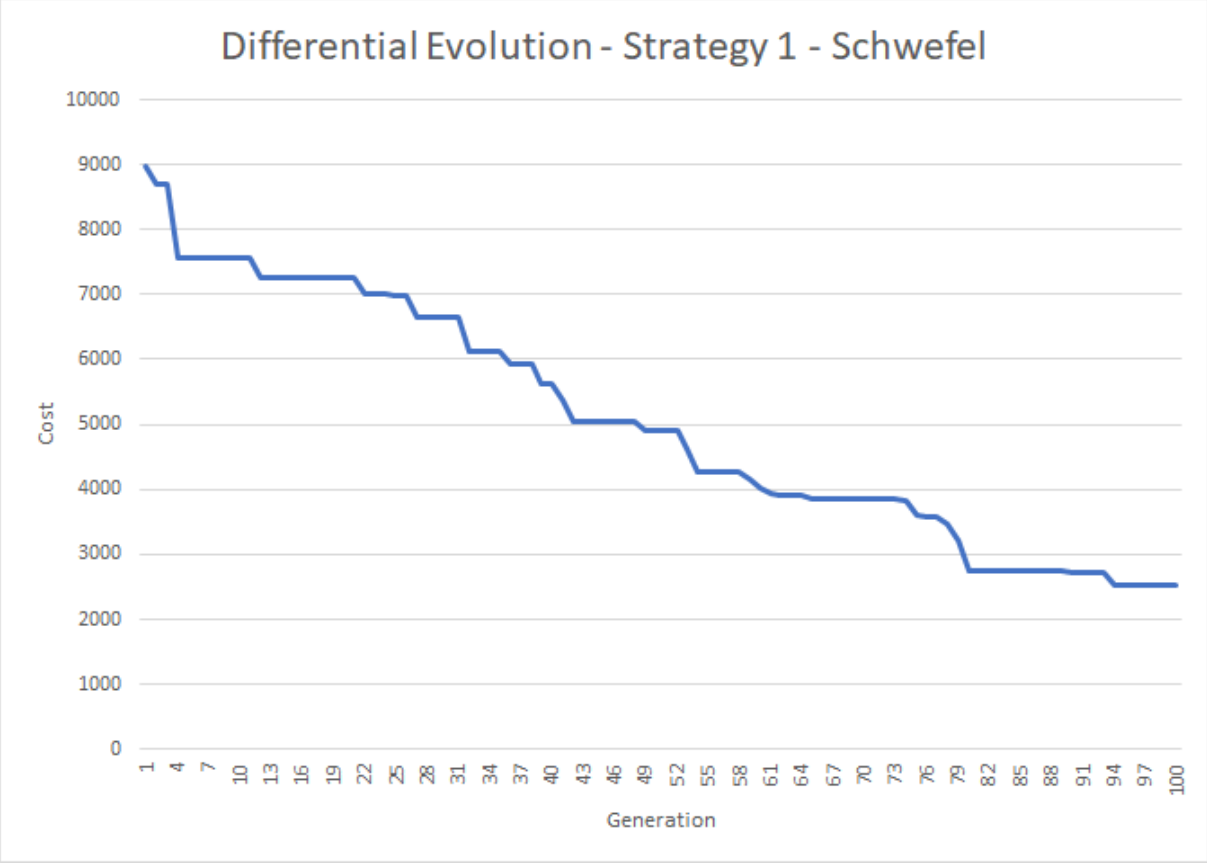

Table 9: Differential Evolution - Strategy 1 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	3572.96	339.41	1772.60	3571.04	2521.12
2	93.76	27.60	151.49	73.14	40.82
3	304950.86	193170.45	1029141.32	263284.03	73813.38
4	-28394.98	4154.84	19349.39	-28171.76	-39278.30
5	1.53	0.13	0.66	1.53	1.25
6	-35.46	0.43	2.06	-35.45	-36.53
7	42.23	1.23	6.36	42.44	38.13
8	-10.62	7.18	27.93	-17.66	-22.41
9	102.49	8.27	37.15	110.22	80.83
10	-12930.67	819.25	3979.40	-12899.64	-15614.14
11	-9375.61	854.25	3521.98	-8821.06	-11226.20
12	5.06	1.04	5.20	4.99	2.63
13	-15.93	0.67	3.14	-15.90	-17.56
14	-4.90	0.58	2.51	-4.83	-6.28
15	43635.68	23424.97	111182.31	40216.23	10654.49
16	1.44	0.33	1.30	1.43	0.88
17	131.86	30.43	145.10	131.08	62.11
18	82.80	13.61	66.82	82.36	41.27

Figure 3: Example of Population Improvement

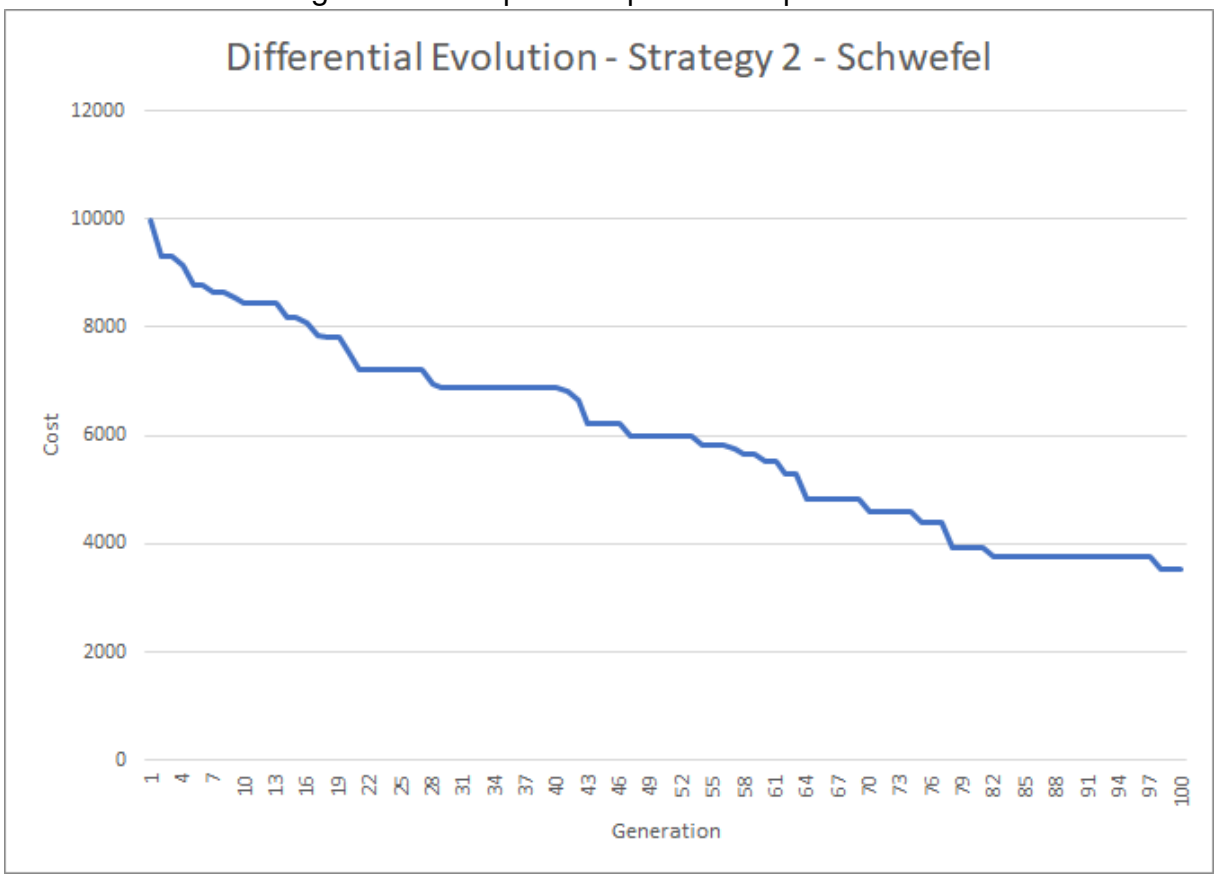

Table 10: Differential Evolution - Strategy 2 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	4176.34	250.09	1463.62	4213.05	3515.16
2	2106.45	369.85	1422.81	2033.97	1441.52
3	60706237.58	25089958.39	145525201.00	59094934.50	17942519.00
4	45957.04	12546.90	52910.44	46006.83	18490.81
5	14.19	2.69	13.68	13.83	9.76
6	-34.00	0.68	3.21	-33.90	-35.88
7	42.87	1.46	6.97	42.95	38.75
8	51.56	9.90	38.93	51.40	31.84
9	234.05	16.42	60.65	235.11	201.34
10	-11918.66	553.32	2846.65	-11824.00	-13815.11
11	-8717.29	517.75	2269.63	-8588.60	-10110.89
12	5.50	0.70	3.24	5.62	4.05
13	-15.09	0.53	2.03	-15.05	-16.24
14	-3.70	0.39	1.80	-3.71	-4.64
15	9192726.38	3019545.14	14144416.50	6962650.50	5130936.50
16	11.12	1.98	8.65	13.15	7.57
17	2229.66	381.48	1907.38	2249.58	1558.29
18	128.79	14.62	71.08	132.05	94.88

Figure 4: Example of Population Improvement

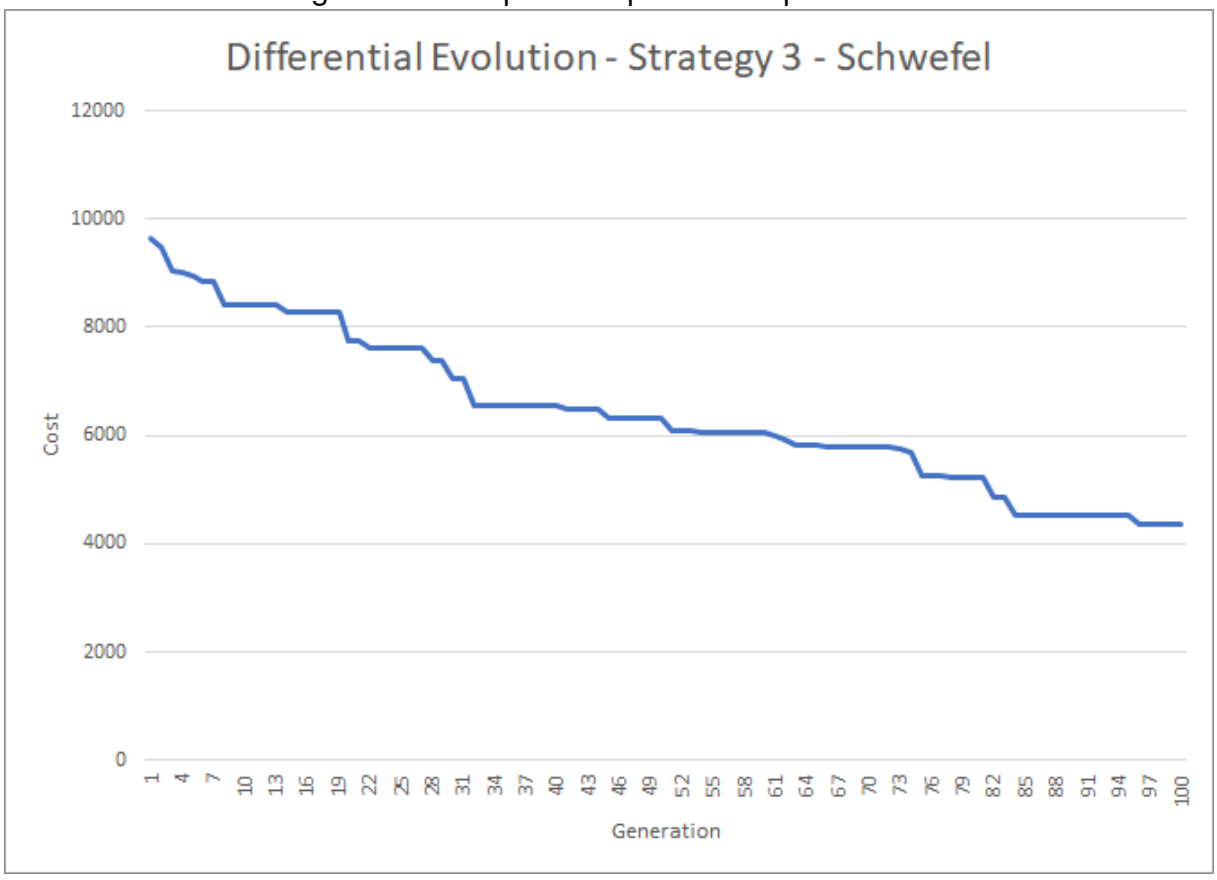

Table 11: Differential Evolution - Strategy 3 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	5424.07	404.52	1779.44	5519.54	4343.56
2	171.58	41.77	227.38	173.46	84.62
3	520226.97	225380.60	1134241.47	481481.70	132536.33
4	-25416.01	4972.87	23562.68	-25361.36	-37853.47
5	2.07	0.24	1.24	2.03	1.54
6	-35.68	0.60	2.24	-35.48	-36.98
7	40.49	1.21	5.39	40.56	37.60
8	0.00	6.08	31.40	4.20	-18.92
9	141.62	11.17	57.68	142.48	115.30
10	-10739.18	593.64	3093.11	-10790.80	-12611.68
11	-7242.74	423.05	1790.35	-7115.45	-8454.09
12	9.82	0.59	2.68	8.91	8.27
13	-16.18	0.64	2.93	-16.26	-17.82
14	-4.56	0.52	2.84	-4.48	-6.44
15	67614.13	31949.76	150805.33	51034.64	24150.13
16	2.28	0.59	3.05	2.16	1.61
17	234.51	40.18	202.12	229.54	136.71
18	88.84	11.21	53.56	86.04	64.68

Figure 5: Example of Population Improvement

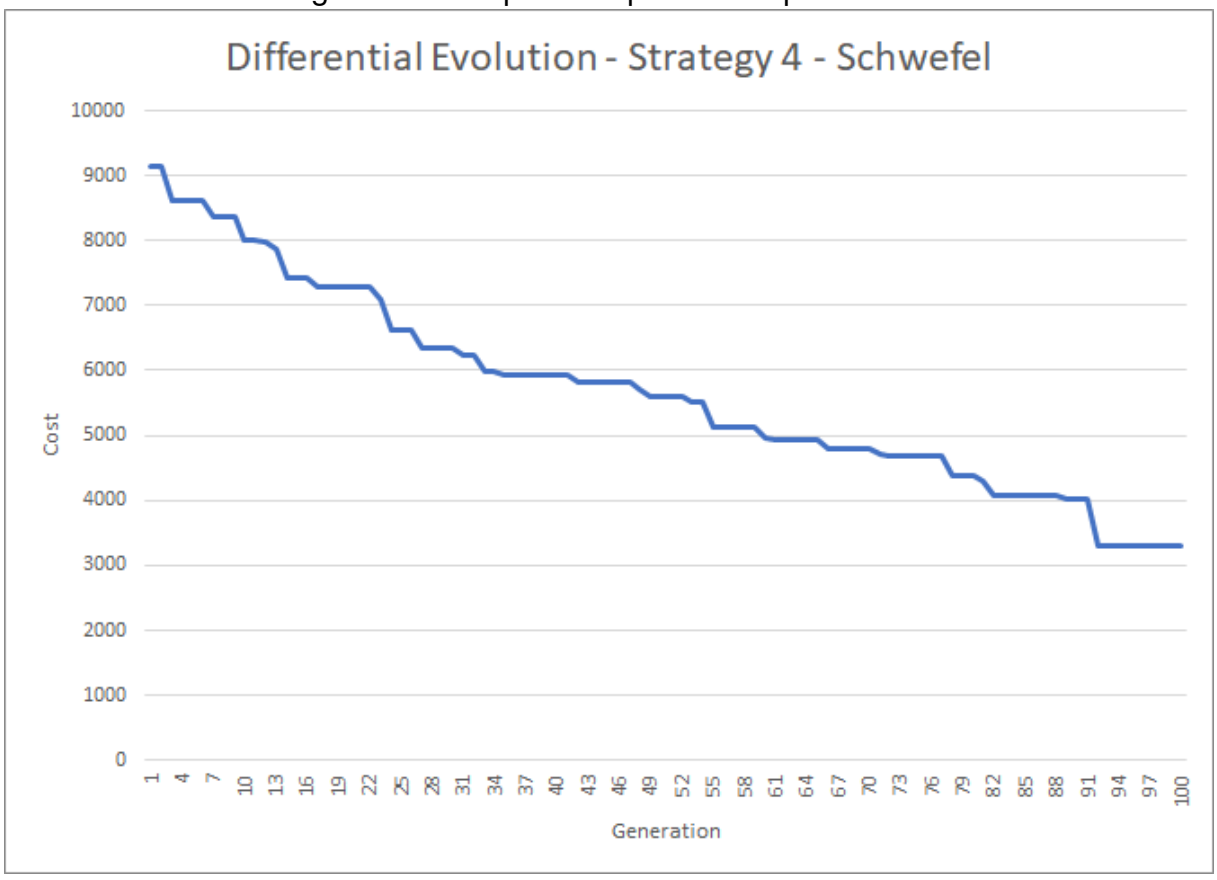

Table 12: Differential Evolution - Strategy 4 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	4192.13	392.61	1587.30	4243.86	3313.34
2	1939.71	407.22	1675.42	1990.82	1190.95
3	57577200.66	22134638.43	91138265.00	55376571.00	14681815.00
4	41970.15	12387.08	54190.87	42919.91	16036.49
5	13.18	2.59	12.07	14.62	8.10
6	-33.56	0.59	2.63	-33.55	-34.80
7	44.07	1.63	7.53	44.46	39.74
8	56.25	8.53	43.05	57.25	35.33
9	262.24	15.28	75.35	262.52	215.76
10	-12277.01	645.52	2731.21	-12285.22	-13844.73
11	-9995.47	922.60	3971.90	-11578.07	-12033.35
12	4.34	0.63	3.28	4.33	2.69
13	-14.65	0.61	3.17	-14.60	-16.70
14	-3.20	0.46	2.21	-3.15	-4.57
15	8658983.20	3325287.75	17543205.70	5876402.75	1773855.30
16	13.50	2.54	12.05	14.49	7.84
17	2137.31	343.10	1411.92	2181.63	1442.91
18	154.28	15.63	70.54	155.88	111.55

Figure 6: Example of Population Improvement

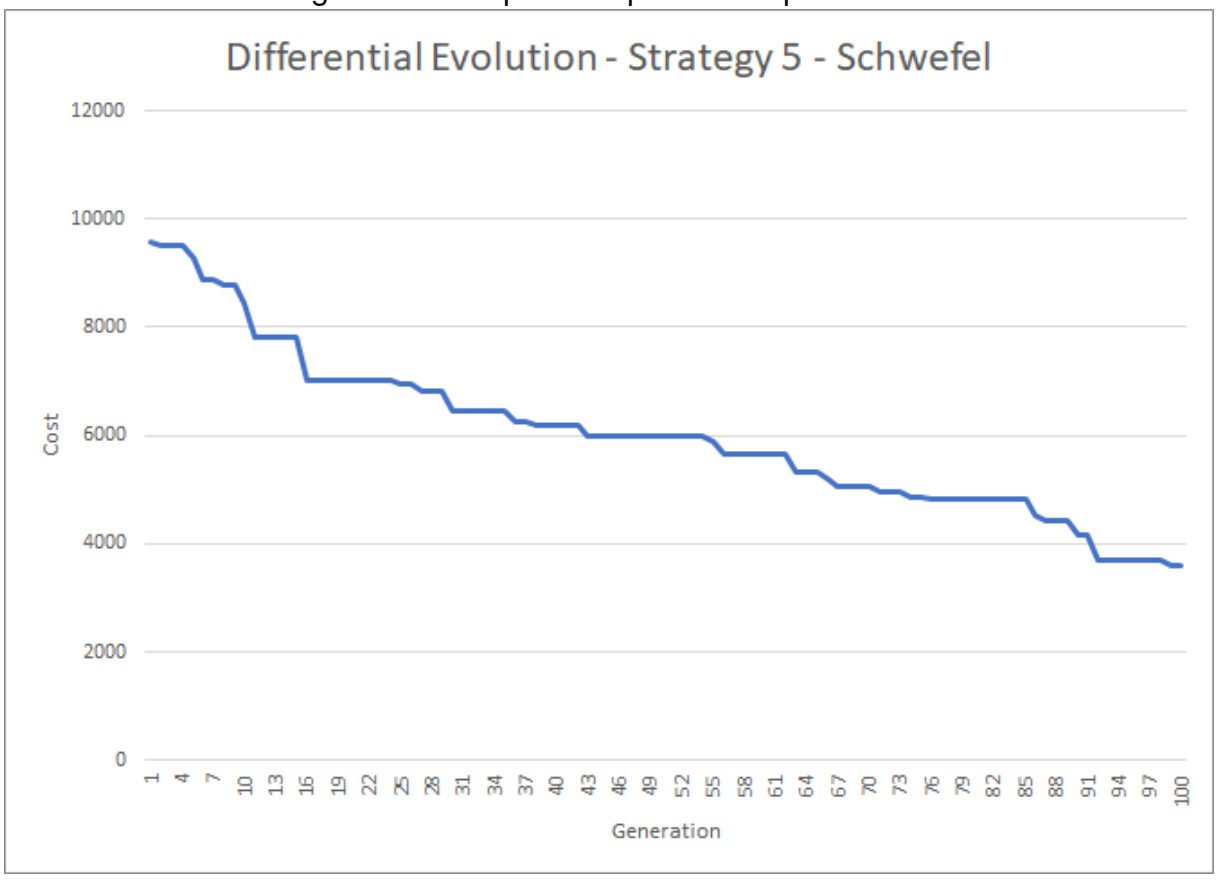

Table 13: Differential Evolution - Strategy 5 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	4073.10	241.71	903.24	4061.62	3590.73
2	6125.98	876.01	3761.74	6205.94	4572.51
3	415663810.20	143597650.58	611217410.00	400226615.00	180894240.00
4	155061.60	29023.18	118753.66	162110.29	87810.94
5	40.54	5.34	21.02	40.92	27.70
6	-32.56	0.56	2.54	-32.48	-33.98
7	44.98	1.57	7.47	45.33	39.75
8	109.62	11.70	51.60	119.46	80.39
9	335.19	17.21	86.10	335.28	276.21
10	-11892.95	547.62	2279.03	-11823.36	-13283.21
11	-9523.54	589.75	2726.33	-9155.69	-11301.84
12	4.60	0.53	2.64	4.51	3.31
13	-14.00	0.42	1.87	-14.01	-14.93
14	-3.05	0.41	2.01	-3.04	-4.16
15	63303827.34	18118195.84	76301744.00	62570476.00	28656686.00
16	26.44	4.46	17.34	25.97	18.40
17	6464.35	1047.03	4526.89	6594.59	3940.67
18	187.97	17.11	70.31	190.53	145.90

Figure 7: Example of Population Improvement

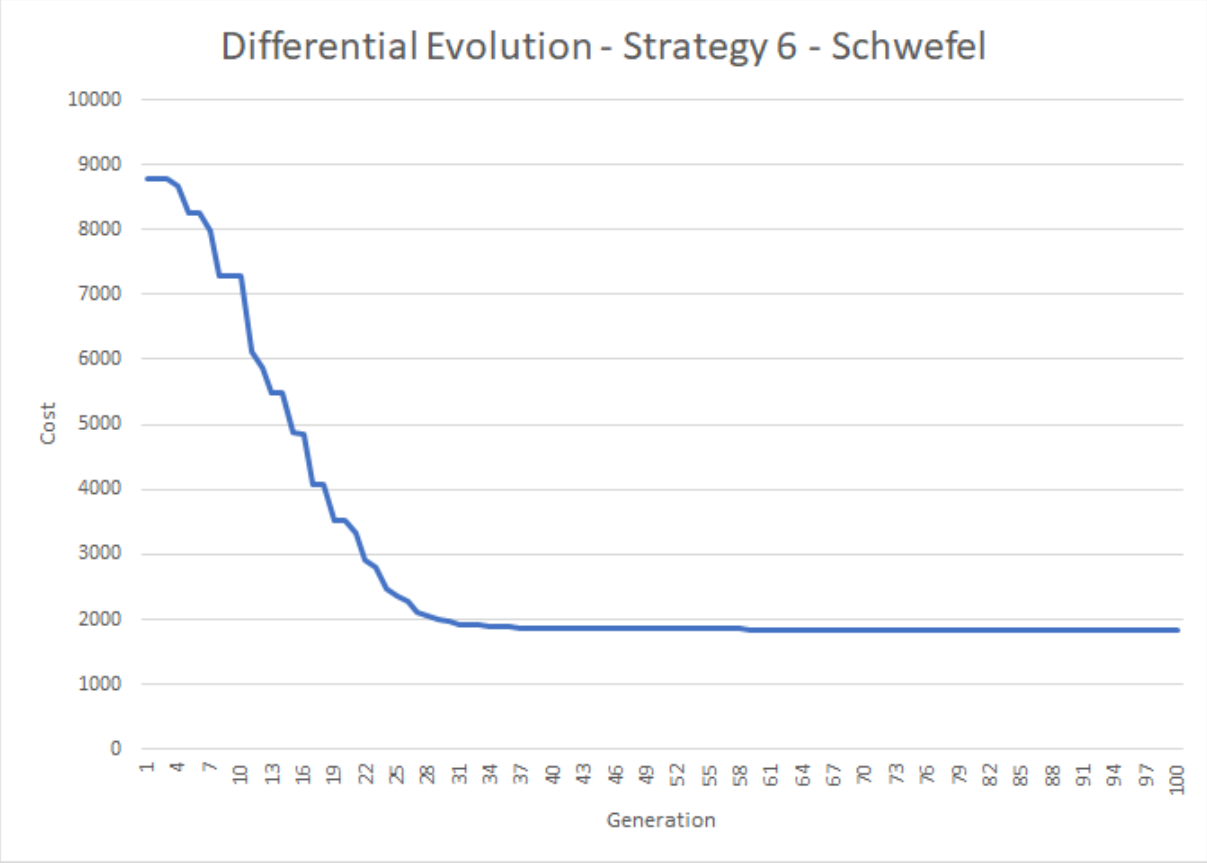

Table 14: Differential Evolution - Strategy 6 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	3029.23	477.49	2247.99	3060.95	1844.78
2	306.41	247.63	1151.03	319.69	25.03
3	7532626.55	9311763.05	42871038.98	18612839.35	122850.02
4	-32180.24	20721.20	96385.15	-42667.28	-65153.50
5	3.00	1.56	7.97	2.59	1.11
6	-31.35	0.95	4.16	-31.45	-33.55
7	53.02	4.18	19.68	53.09	43.76
8	-17.02	20.80	96.62	1.18	-58.33
9	176.72	41.49	192.04	191.38	69.28
10	-15502.46	1187.81	6224.95	-15660.11	-18780.43
11	-9276.44	1944.66	6890.17	-5520.44	-11769.03
12	7.66	1.29	6.11	7.41	4.88
13	-14.14	2.17	12.23	-13.94	-23.35
14	-4.60	0.81	4.02	-4.65	-6.41
15	1262775.76	1313474.84	5711614.82	302328.54	20456.68
16	8.09	3.52	13.69	6.03	1.46
17	373.91	351.80	1896.50	335.90	50.66
18	128.09	67.46	444.70	153.85	13.36

Figure 8: Example of Population Improvement

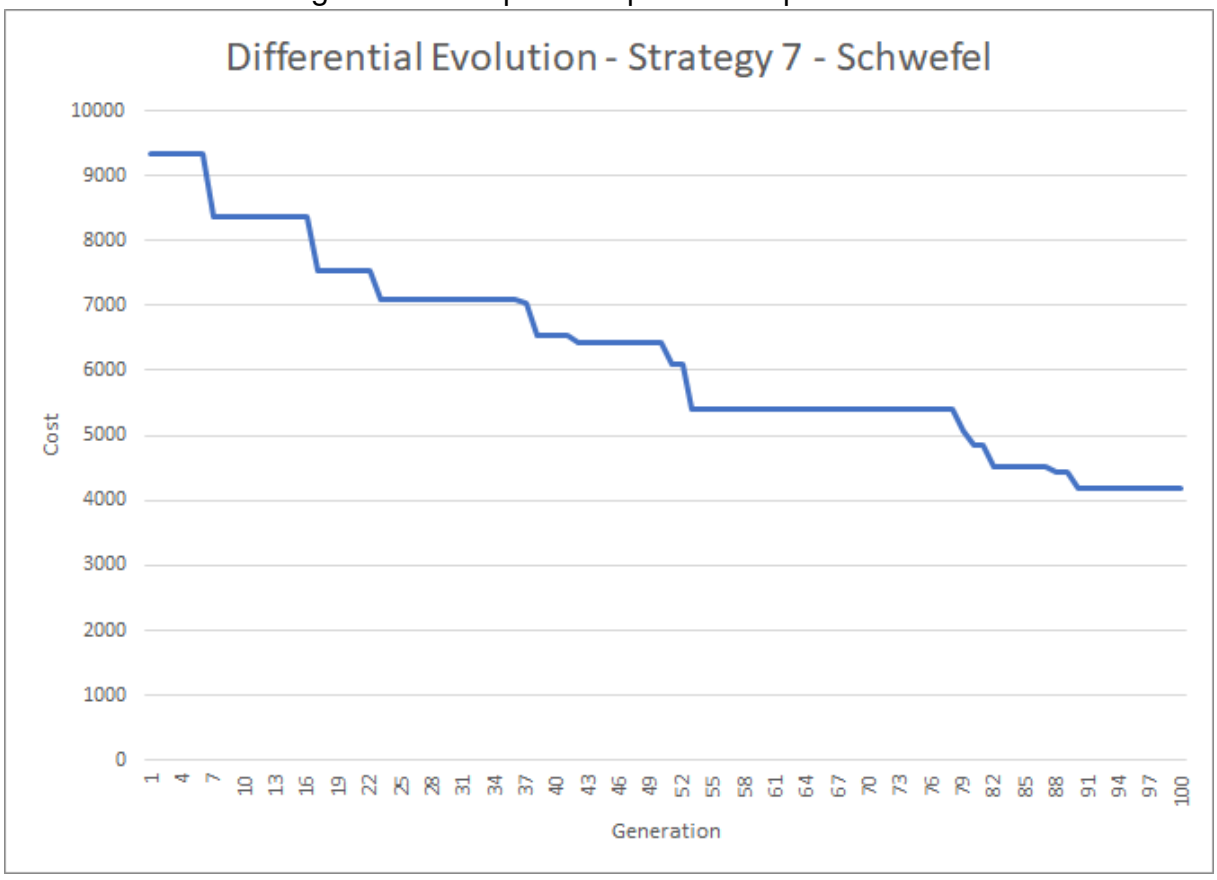

Table 15: Differential Evolution - Strategy 7 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	5693.34	639.33	2931.34	5684.20	4183.22
2	1353.85	443.31	2224.90	1607.23	461.74
3	50554471.46	30957288.00	145195741.00	35339654.50	14822389.00
4	23265.08	12630.05	49691.98	27855.05	-2143.00
5	9.24	2.35	11.71	6.50	5.21
6	-30.50	0.66	3.37	-30.45	-32.88
7	58.48	2.86	11.28	59.08	51.26
8	56.25	13.49	57.47	56.87	31.11
9	272.92	30.08	146.32	275.85	188.40
10	-8197.20	616.54	2981.19	-8221.45	-9806.17
11	-7244.47	1039.76	5829.04	-7215.46	-10669.33
12	4.90	0.70	3.26	4.82	3.00
13	-10.75	0.67	3.20	-10.74	-12.70
14	-2.46	0.31	1.39	-2.45	-3.40
15	7683132.52	4684608.87	27364036.80	5515877.30	2021164.20
16	11.40	3.83	19.68	22.53	5.53
17	1440.35	399.31	1583.50	1666.24	825.20
18	181.88	31.73	116.69	179.65	133.10

Figure 9: Example of Population Improvement

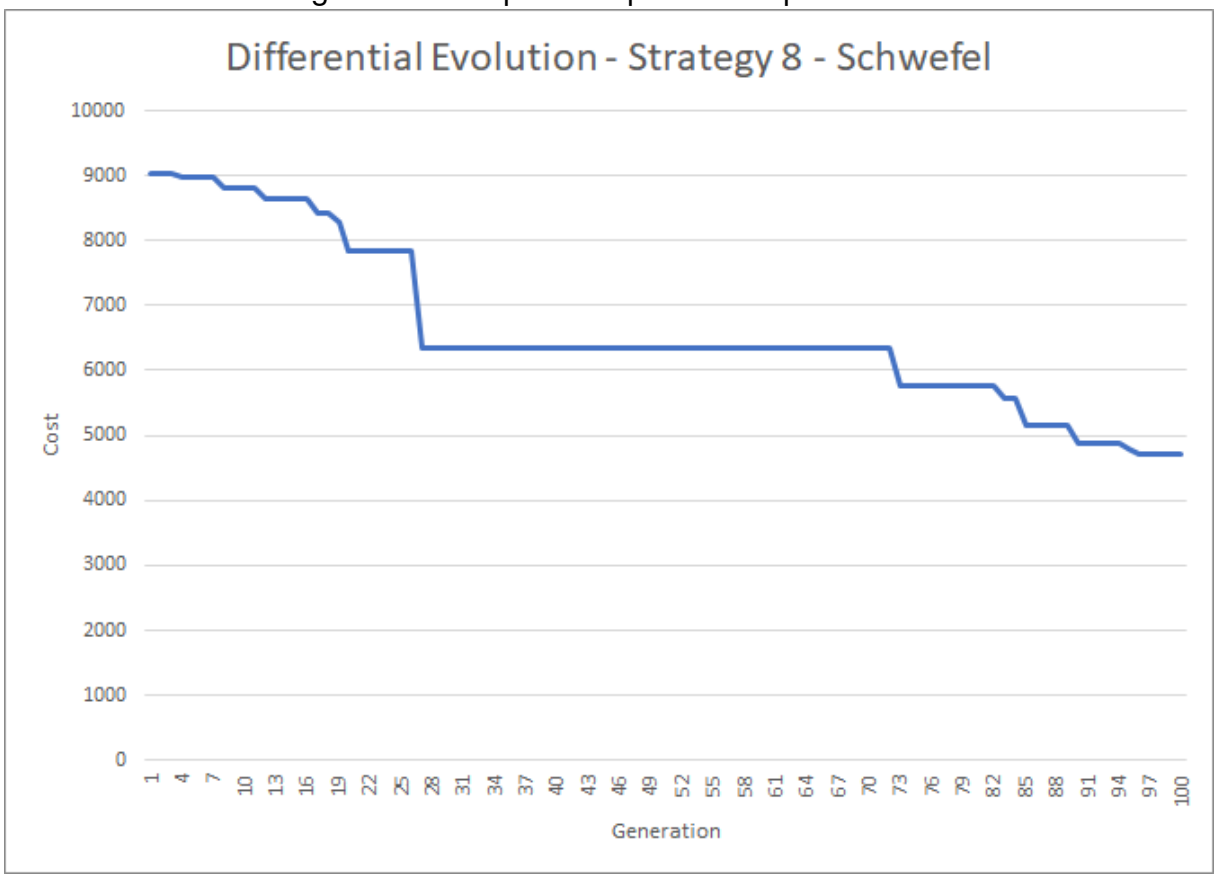

Table 16: Differential Evolution - Strategy 8 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	6531.14	908.86	3526.90	6750.97	4712.35
2	186.03	136.14	677.47	320.52	5.82
3	2103049.60	2857073.84	12960229.27	3009385.50	22633.73
4	-13408.29	11631.20	49777.96	-26707.32	-36971.04
5	2.34	1.01	4.05	2.10	1.08
6	-33.08	0.67	2.76	-33.00	-34.77
7	50.33	2.37	8.04	50.26	46.42
8	-41.49	24.77	109.72	-49.72	-72.52
9	79.28	32.67	156.69	63.62	25.79
10	-8492.33	1069.25	4132.24	-8078.64	-11109.90
11	-5818.92	995.01	4061.75	-5732.20	-8226.33
12	7.95	1.62	6.47	7.42	5.80
13	-12.33	0.59	2.45	-12.30	-13.49
14	-3.20	0.42	1.69	-3.13	-4.18
15	302240.38	401037.01	2165953.36	322340.66	16669.45
16	2.40	1.29	5.66	2.30	0.30
17	235.75	148.96	815.06	256.21	30.19
18	73.82	26.81	113.09	61.20	29.03

Figure 10: Example of Population Improvement

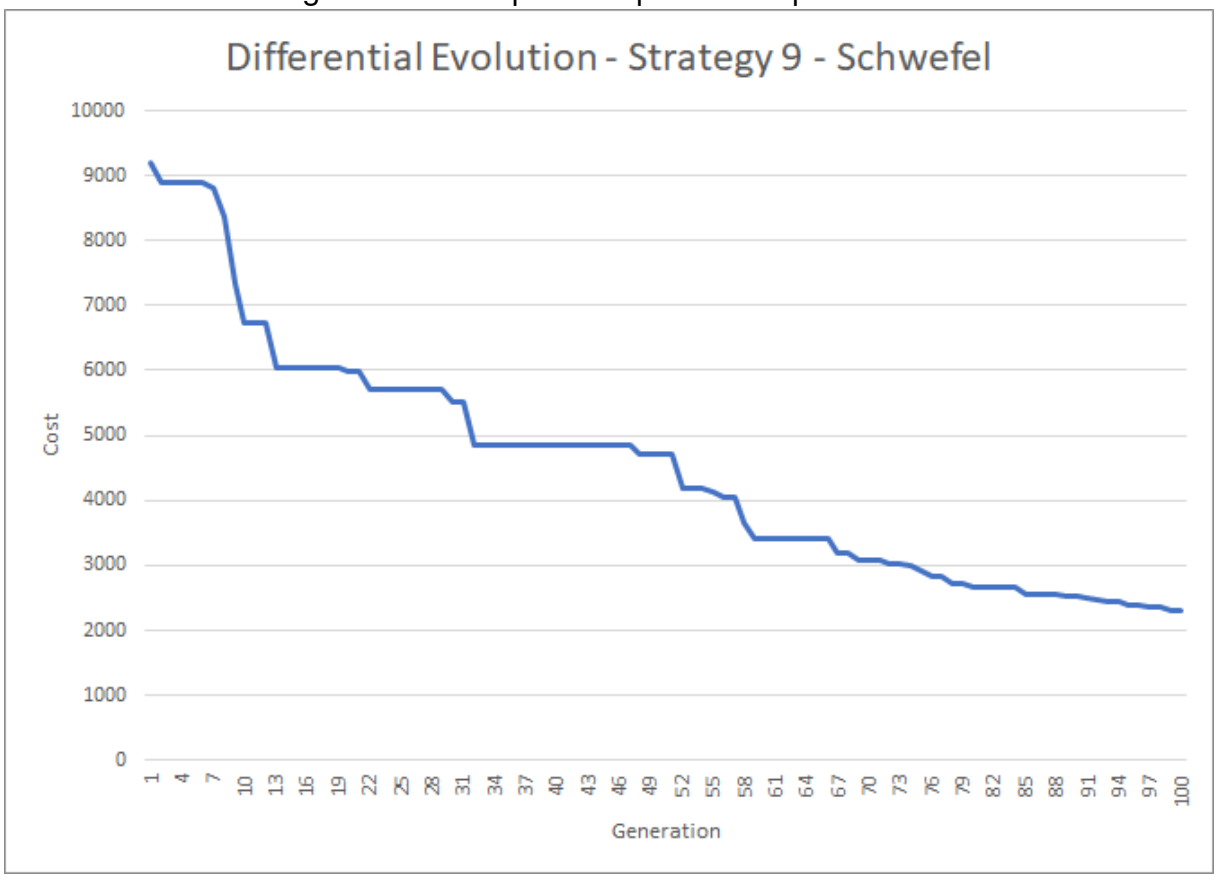

Table 17: Differential Evolution - Strategy 9 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	4179.91	1387.25	4886.81	4216.82	2302.35
2	257.29	135.01	731.39	242.11	92.94
3	5746870.35	4827320.10	17758973.62	3226821.10	365574.38
4	4270.27	10127.99	48362.91	1486.72	-20156.19
5	2.53	0.74	2.82	2.29	1.34
6	-29.96	0.88	4.79	-30.09	-32.00
7	60.61	3.22	15.38	61.11	52.74
8	52.99	16.70	74.48	50.31	20.25
9	216.34	40.30	217.91	215.11	122.58
10	-9473.90	1548.74	8380.76	-8640.45	-15273.80
11	-8133.00	1758.73	8008.61	-7305.37	-12770.06
12	5.90	1.01	5.00	5.80	3.70
13	-10.38	0.79	3.41	-11.39	-12.33
14	-2.44	0.26	1.20	-2.44	-3.04
15	630700.49	547822.78	2459569.56	338720.82	21906.94
16	19.97	13.78	60.58	31.48	4.79
17	342.41	156.23	736.89	349.73	94.02
18	235.02	55.49	265.26	240.92	104.43

Figure 11: Example of Population Improvement

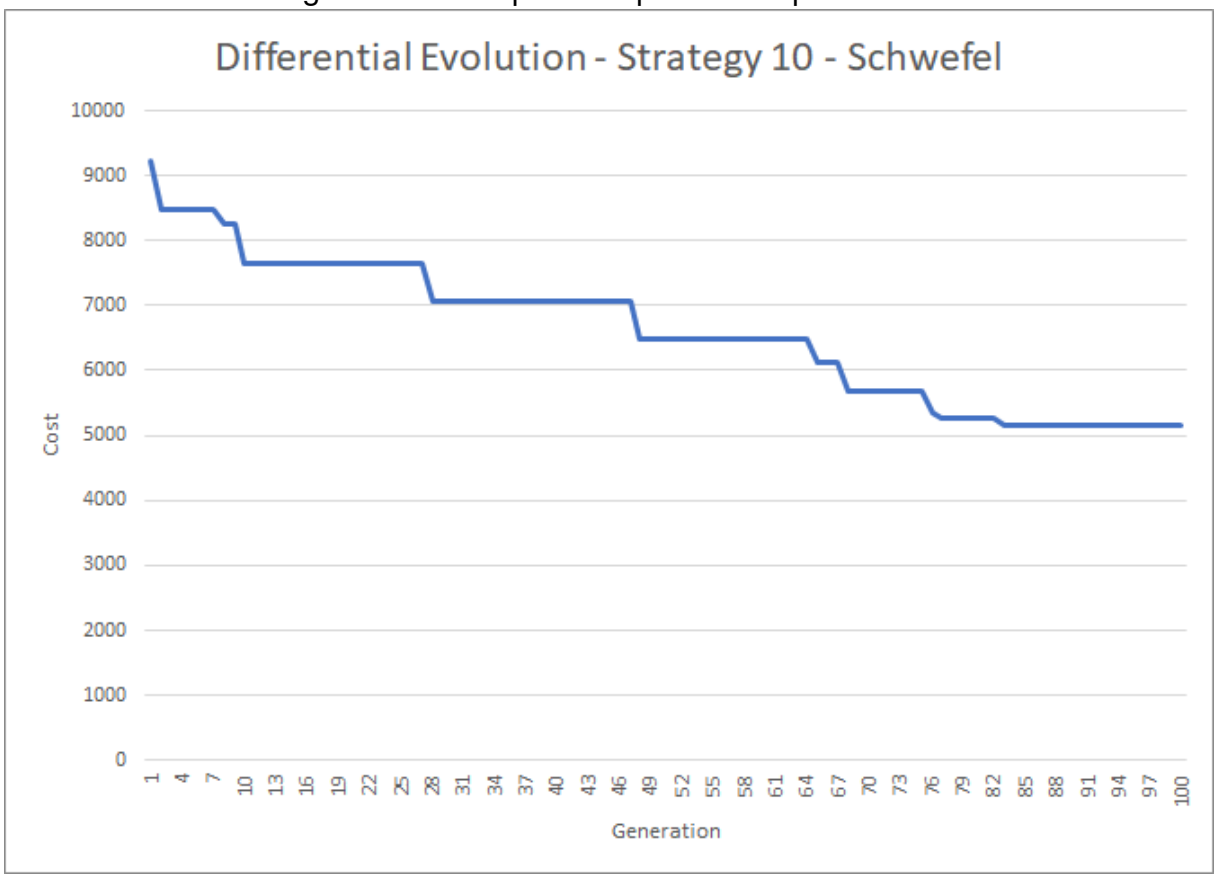

Table 18: Differential Evolution - Strategy 10 - 50 Experiments

F(x)	Average	Std. Deviation	Range	Median	Min
1	6101.30	455.98	1737.94	6210.21	5164.84
2	19566.82	2623.28	9797.18	20015.63	14504.13
3	4528202408.00	1419008590.65	6202405000.00	4529514600.00	1654435000.00
4	513991.39	87496.10	371162.32	513430.99	319251.63
5	128.80	20.54	93.64	134.45	83.26
6	-27.65	0.57	2.89	-27.59	-29.35
7	65.16	2.74	13.30	65.45	56.81
8	237.99	24.95	114.29	239.75	173.44
9	457.96	11.25	51.91	459.14	433.93
10	-7852.74	697.86	3357.99	-7661.51	-10074.07
11	-7845.51	912.14	4226.30	-7852.78	-10009.34
12	5.15	1.12	7.00	5.33	0.50
13	-9.69	0.72	3.77	-9.29	-12.33
14	-2.05	0.21	0.90	-2.04	-2.58
15	611982508.20	192241588.39	764983860.00	561385045.00	245587540.00
16	89.77	16.44	85.47	74.61	43.93
17	19482.21	2972.64	15177.70	19682.30	13070.13
18	388.59	28.58	134.60	394.52	302.49

6 Analysis

First lets look compare our local search results in Table 7 to our genetic algorithm results in table Table 8. The interesting thing here is that while most average results are better in the genetic algorithm, local search did perform better for a few functions. Specifically, functions 2, 5, and 17 performed better with local search and thus had a lower fitness value. Another interesting observation is that function 4 seems to be to be producing invalid results since in the genetic algorithm we are getting a very large negative number when the optimal minimum for that function should be 0. Therefore we need to verify that function 4 is implemented correctly in our program.

For differential evolution we tested 10 different mutation and crossover strategies to compare and contrast which strategies worked best for which functions. The analysis here is a little more complicated. First, when in general when you compare the genetic algorithm results to the different differential evolution strategy results, you will see that it's a little mixed as far as which one performs better. For many functions, one of the differential evolution strategies has produced better results than the comparable genetic algorithm, but not for all functions. For example, looking at the Schwefel function in Table 8, the genetic algorithm found an average fitness of 1409.28 and minimum fitness of 699.11. In contrast, the best this function performed using differential evolution can be found in Table 14, using strategy 6, where the DE algorithm produced an average fitness of 3029.23 and a minimum fitness of 1844.78 using the Schwefel function.

For most functions however the differential evolution algorithm did produce better results in one strategy or another over the genetic algorithm. If you find the best performing functions (on average) for differential evolution, you see these include:

Function 2, Strategy 1
Function 3, Strategy 1
Function 4, Strategy 6
Function 5, Strategy 1
Function 8, Strategy 8
Function 9, Strategy 8
Function 10, Strategy 6
Function 11, Strategy 4
Function 12, Strategy 4
Function 15, Strategy 1
Function 16, Strategy 1
Function 17, Strategy 1

So as you can see, out of the 18 functions, 12 performed best using one of the differential evolution strategies. Likewise, functions 1, 6, 7, 13, 14, and 18 performed better using the genetic algorithm. DE strategy 1 performed the best on average, with strategies 3, 6, and 8 all tied for second place.

7 Conclusions

After our analysis it should be clear that our results are a bit mixed. For a few functions, local search performed the best. Especially with functions 2 and 5. Neither the genetic algorithm, nor the differential evolution algorithm could surpass the local search in average best fitness. The differential algorithm did get fairly close to the local search with function 5 using strategy 1, where the average fitness was 1.53 versus local search's average of 1.0. In general though, the genetic algorithm and differential algorithm did perform the best so far out of all previous experiments, the latter depending on which strategy was used. From our testing it appears that the best performing DE strategies were 1, 4, 6, and 8. None of the other strategies produced best average results for any of the 18 functions. Therefore, in future testing we recommend further optimizing these strategies to improve our results.

Performance wise, the genetic algorithm and differential evolution algorithm require far less execution time than local search, while producing on average better results. Unlike local search, there was not a huge variation in execution times between the different functions. In addition, the minimum and maximum execution time range is smaller on average than local search as well. In conclusion, despite some mixed results for a couple functions, these evolutionary algorithms do perform the best out of all experiments conducted so far.