Central Washington University

CS471 Optimization

Winter 2020

Project3: Genetic Algorithm and Differential Evolution

Student:
Chao Huang Lin
chao.huanglin@cwu.edu

Professor:
Dr. Donald Davendra
Donald.Davendra@cwu.edu

February 11, 2020

Contents

1 Introduction

For this project, two optimization algorithms will be tested. These are Genetic Algorithm (GA) and Differential Evolution Algorithm (DE).

The GA is a heuristic search and optimization techniques that simulate the process of natural evolution. The main operations of GA are: Selection, Crossover, Mutation and Elitism.

The DE algorithm employs the difference of two randomly selected parameter vectors as the source fo random variations for a third parameter vector. The some advantages of DE are: few numbers of control parameters which make it easier to calibrate or tune, it is inherently parallel, and it has a faster convergence. There many strategies of DE, in this project 10 different strategies will be tested.

2 Computer Specification

For this Lab, we used a Microsoft Surface Pro 3 which has the following specification: Core i7-4650U CPU 1.7GHz 2.3GHz with 8 GB of RAM.

The C++ codes have been run on docker where 2 CPUs and 2GB of RAM was assigned. The Docker image was from professor Szilard "szilardvajda/ubuntu_cs470"

3 Method

The GA and DE algorithms are coded using C++ object oriented programming,

python code to run and display result in table and figures.

50 runs of each algorithms, the population size is 200, and the generations or iterations is 100, number of dimensions is 30.

Listing 1: matrix_double_pointer class

```
class matrix_double_pointer
{
int m; // mxm matrix
long double **matrix;

public:
matrix_double_pointer(int m); // constructor
matrix_double_pointer(const matrix_double_pointer& a); // copy constructor for return
~matrix_double_pointer(); //destructor
void init_identity(); //init identity matrix
void init(long double x); //init matrix all with x
void init_rand(); //init matrix with pseudo random
void print_values();
void print_memory_address();
matrix_double_pointer operator* (const matrix_double_pointer& a);
matrix_double_pointer operator^ (int n);
matrix_double_pointer& operator= (const matrix_double_pointer& a); // assigment
long double** get_matrix();
};
```

To implement classes that work with pointers, we must take some special consideration [1]. These classes should have:

- a destructor (to delete the allocated memory).
- a copy constructor (to copy the object when is returned, passed as parameter or initialization from another object of the same class).
- an overloaded operator = (to assign or copy the data to another object).

From listing 1 and 2 we can observe that all these 3 member functions are implemented. In addition, we have overload the operator $\wedge$ to get the matrix power in a more intuitive way.

Listing 2: matrix_linked_list class

```
struct node
{
    long double data;
    int j;
    node *right;
};
struct head_node
{
    long double data;
    int i;
    int j;
    node *right;
    head_node *down_head_node;
};

class matrix_linked_list
{
    int m; // mxm matrix
    head_node *main_head_node;
    head_node *current_head_node; //for get_value_loop_opt()
    node *current_node; //for get_value_loop_opt()
    head_node *current_head_node_add; // add_value_loop_opt()
    node *current_node_add; // for add_value_loop_opt()

    void start_using_get_value_loop_opt();
    long double get_value_loop_opt(int i, int j); // optimized version of get_value (make the loop
        faster using temp node address)
    void add_value_loop_opt(long double value, int i, int j); // // optimized version of add_value
        (make the loop faster using temp node address)

    public:
    matrix_linked_list(); // constructor
    matrix_linked_list(int m); // constructor
    matrix_linked_list(const matrix_linked_list& a); // copy constructor for return
    ~matrix_linked_list(); //destructor
    void load(long double **matrix, int m); // receive square matrix and size mxm
    void delete_all_node();
    void print_values();
    void print_values_ij(); //print the data and also print i,j
    void set_main_head_node(head_node *main_head_node);
    long double get_value(int i, int j);
    void add_value(long double value, int i, int j); // add value to the existing i,j if not ->
        create new node
```

```
    matrix_linked_list operator^ (int n);
    matrix_linked_list& operator= (const matrix_linked_list& a); // assigment
};
```

4 Important Notes

During the development of this project there were many problems that need to be fixed in order to get optimal results:

1) The mutation of GA and the trial of DE can make the data go outside of the range, to solve this problem a function is coded to truncate the values in the range.

2) The project has been run in single processor mode and parallel multiprocessor mode. The results presented in this document are only from single processor mode since there was some problem with CPU clock time in parallel mode. According to [2] the clock() function in C++ measures the CPU time used by the entire program so other processes or threads that are not part of the algorithm that need measurement also get counted. (This problem might be caused by Python subprocess package)

3) Some DE algorithms did not converge, the cost was very high. This is caused by the parameters which are not calibrated, to solve this problem, the parameters of different algorithm of Differential Evolution had been calibrated manually one by one.

   the calibrated parameters are:

5 Results

The results of applying GA and DE optimization algorithms with 18 benchmarking functions are displayed in the following pages:

5.1 Function 1: Schwefel

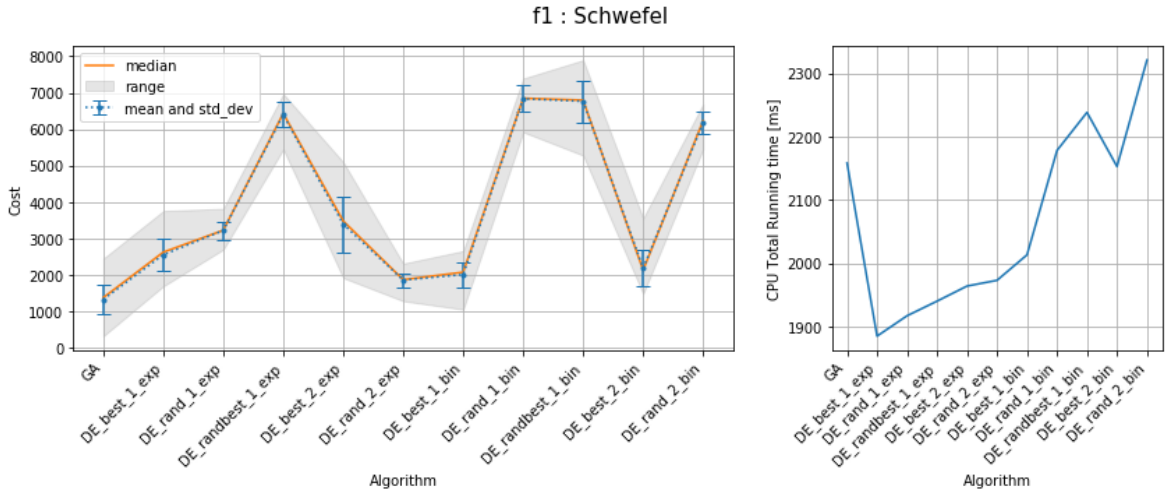

Figure 1: Cost and CPU total running time of Function 1: Schwefel

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	1330.170	388.519	1379.190	322.844	2451.800	2158.370
DE_best_1_exp	2554.650	454.357	2625.350	1692.530	3758.570	1885.140
DE_rand_1_exp	3221.480	245.530	3226.450	2698.210	3818.370	1917.270
DE_randbest_1_exp	6393.850	341.369	6432.170	5447.420	6967.870	1940.380
DE_best_2_exp	3393.730	766.011	3481.430	1919.550	5097.510	1964.200
DE_rand_2_exp	1859.860	190.640	1870.870	1285.870	2314.850	1973.270
DE_best_1_bin	2012.120	343.998	2076.640	1055.310	2660.160	2013.510
DE_rand_1_bin	6833.980	360.550	6847.770	5919.700	7390.060	2178.320
DE_randbest_1_bin	6764.300	575.049	6794.490	5279.800	7895.460	2238.470
DE_best_2_bin	2187.930	488.011	2130.950	1484.130	3557.450	2153.020
DE_rand_2_bin	6168.990	298.901	6230.760	5457.080	6689.420	2321.140

Table 1: Function 1: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 1330.170000
best DE cost (mean): 1859.860000
best DE: DE_rand_2_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_rand_2_exp then **GA is better than DE_rand_2_exp**.

5.2 Function 2: De Jong 1

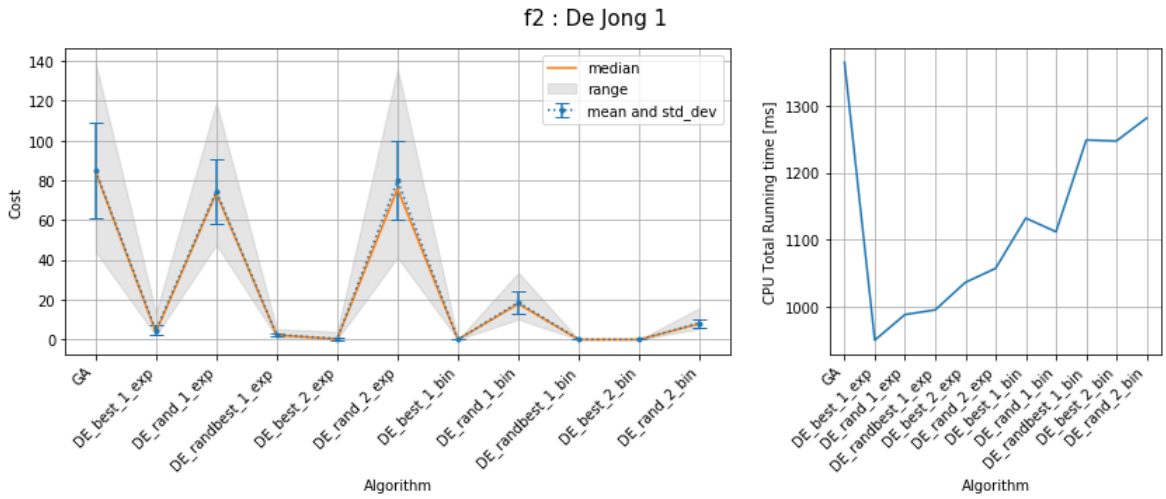

Figure 2: Cost and CPU total running time of Function 2: De Jong 1

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	85.044	23.894	84.611	44.305	139.288	1365.260
DE_best_1_exp	4.529	2.486	3.701	2.033	13.354	949.713
DE_rand_1_exp	74.558	16.062	73.572	47.658	119.195	988.005
DE_randbest_1_exp	2.406	0.862	2.240	1.051	5.319	994.983
DE_best_2_exp	0.459	0.782	0.171	0.013	3.913	1036.180
DE_rand_2_exp	79.866	19.786	75.825	41.275	136.236	1057.180
DE_best_1_bin	0.004	0.004	0.003	0.001	0.019	1132.280
DE_rand_1_bin	18.665	5.393	17.922	10.120	33.667	1111.860
DE_randbest_1_bin	0.002	0.002	0.002	0.001	0.015	1249.120
DE_best_2_bin	0.007	0.005	0.006	0.002	0.023	1247.720
DE_rand_2_bin	8.319	2.081	7.912	5.392	15.903	1281.980

Table 2: Function 2: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 85.044300
best DE cost (mean): 0.002287
best DE: DE_randbest_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_randbest_1_bin
confidence interval = 95%
p value: 0.000000
DE_randbest_1_bin obtains lower cost than GA then **DE_randbest_1_bin is better than GA**.

5.3 Function 3: Rosenbrok's Saddle

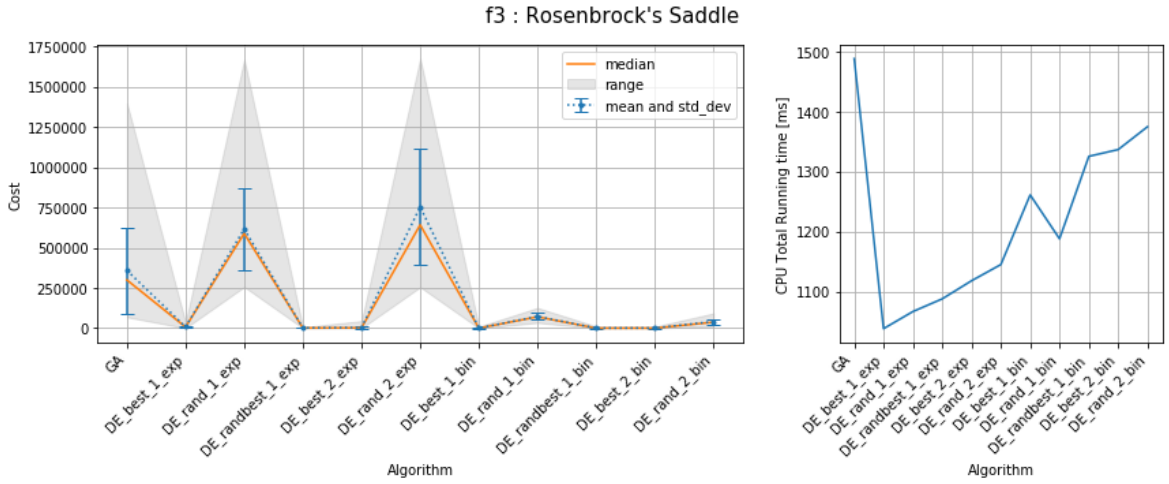

Figure 3: Cost and CPU total running time of Function 3: Rosenbrok's Saddle

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	358923.000	267559.000	299684.000	68740.900	1398620.000	1488.010
DE_best_1_exp	8366.080	7368.590	6825.100	896.082	36143.000	1037.970
DE_rand_1_exp	616375.000	254829.000	588666.000	256876.000	1669420.000	1066.480
DE_randbest_1_exp	1634.280	1297.190	1187.020	359.236	6231.420	1087.650
DE_best_2_exp	3820.620	6851.230	1888.900	128.036	45150.700	1117.830
DE_rand_2_exp	753439.000	361520.000	642237.000	255458.000	1675010.000	1144.840
DE_best_1_bin	520.354	1468.460	124.942	28.947	10194.600	1260.940
DE_rand_1_bin	72972.600	22329.200	70409.300	33871.500	126730.000	1187.770
DE_randbest_1_bin	562.328	1690.900	139.163	32.807	11575.100	1325.200
DE_best_2_bin	1048.130	2325.130	199.843	34.386	10112.200	1336.470
DE_rand_2_bin	41223.200	17445.900	37140.600	11675.300	92562.200	1374.570

Table 3: Function 3: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 358923.000000
best DE cost (mean): 520.354000
best DE: DE_best_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_bin
confidence interval = 95%
p value: 0.000000
DE_best_1_bin obtains lower cost than GA then **DE_best_1_bin is better than GA**.

5.4 Function 4: Rastrigin

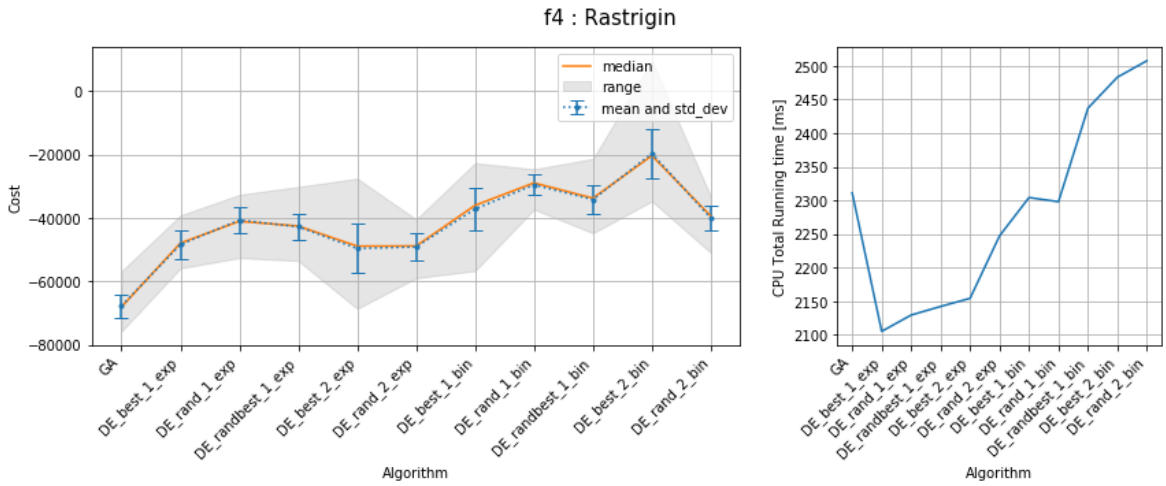

Figure 4: Cost and CPU total running time of Function 4: Rastrigin

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-67715.300	3609.350	-68065.100	-75769.800	-56733.400	2311.080
DE_best_1_exp	-48389.100	4409.680	-47912.100	-55862.600	-39156.300	2105.140
DE_rand_1_exp	-40661.400	4103.810	-41029.900	-52576.200	-32624.400	2129.370
DE_randbest_1_exp	-42764.300	4024.470	-42537.300	-53514.800	-30143.200	2142.170
DE_best_2_exp	-49615.600	7726.570	-48882.600	-68621.700	-27517.200	2154.290
DE_rand_2_exp	-49116.300	4254.270	-48819.600	-58954.800	-40404.200	2247.240
DE_best_1_bin	-37136.400	6739.330	-36004.300	-56677.300	-22626.200	2304.170
DE_rand_1_bin	-29531.800	3221.190	-28905.400	-37277.600	-24578.200	2298.010
DE_randbest_1_bin	-34185.400	4381.230	-33735.800	-44761.300	-21279.500	2437.100
DE_best_2_bin	-19621.900	7905.000	-20318.700	-34698.100	9621.470	2483.290
DE_rand_2_bin	-40013.100	4080.260	-39385.600	-50957.300	-33198.600	2507.650

Table 4: Function 4: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -67715.300000
best DE cost (mean): -49615.600000
best DE: DE_best_2_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_2_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_best_2_exp then **GA is better than DE_best_2_exp**.

5.5 Function 5: Griewangk

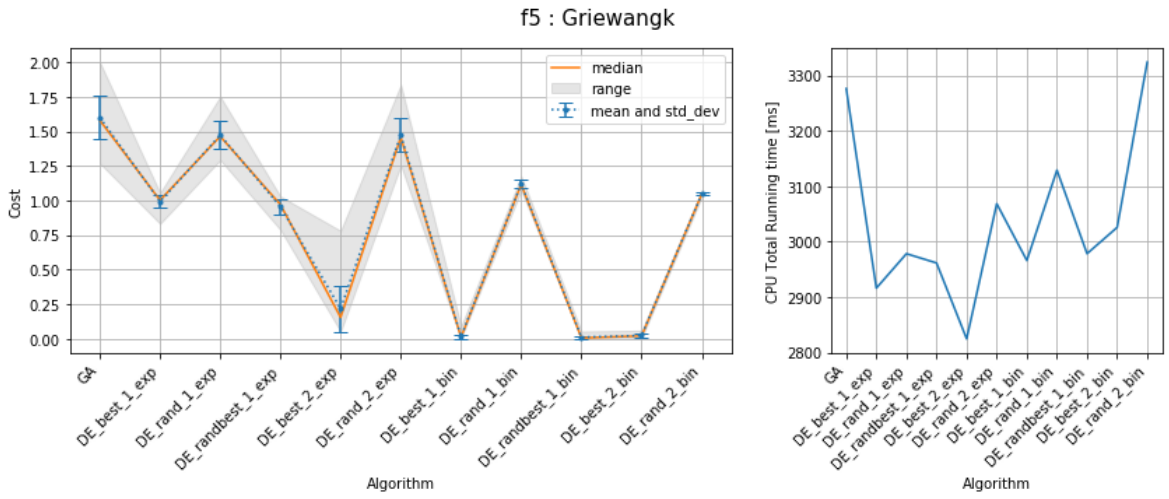

Figure 5: Cost and CPU total running time of Function 5: Griewangk

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	1.603	0.160	1.581	1.270	2.002	3276.470
DE_best_1_exp	0.993	0.048	1.002	0.833	1.057	2916.120
DE_rand_1_exp	1.474	0.103	1.465	1.293	1.754	2978.280
DE_randbest_1_exp	0.956	0.058	0.971	0.789	1.031	2961.530
DE_best_2_exp	0.217	0.169	0.154	0.052	0.782	2824.540
DE_rand_2_exp	1.475	0.120	1.458	1.255	1.839	3068.560
DE_best_1_bin	0.015	0.016	0.011	0.001	0.091	2966.040
DE_rand_1_bin	1.121	0.028	1.117	1.073	1.177	3129.120
DE_randbest_1_bin	0.011	0.011	0.006	0.002	0.054	2978.360
DE_best_2_bin	0.023	0.014	0.020	0.004	0.059	3025.790
DE_rand_2_bin	1.048	0.011	1.047	1.027	1.076	3324.260

Table 5: Function 5: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 1.602620
best DE cost (mean): 0.010972
best DE: DE_randbest_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_randbest_1_bin
confidence interval = 95%
p value: 0.000000
DE_randbest_1_bin obtains lower cost than GA then **DE_randbest_1_bin is better than GA**.

5.6 Function 6: Sine Envelope Sine Wave

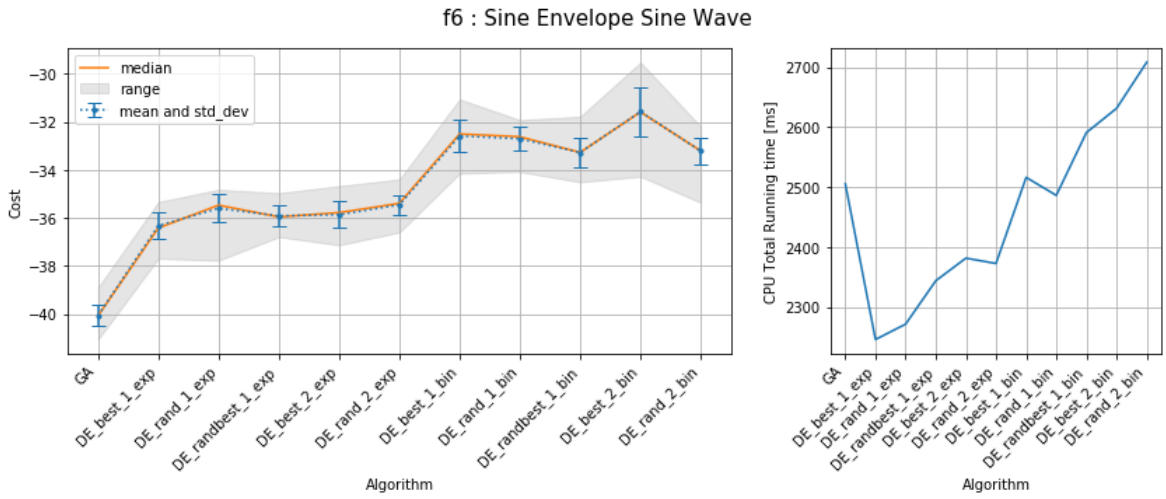

Figure 6: Cost and CPU total running time of Function 6: Sine Envelope Sine Wave

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-40.070	0.437	-40.085	-41.066	-38.894	2505.260
DE_best_1_exp	-36.327	0.565	-36.422	-37.696	-35.335	2245.670
DE_rand_1_exp	-35.602	0.593	-35.471	-37.775	-34.823	2271.290
DE_randbest_1_exp	-35.918	0.431	-35.960	-36.793	-34.967	2343.630
DE_best_2_exp	-35.868	0.547	-35.782	-37.137	-34.672	2381.410
DE_rand_2_exp	-35.457	0.422	-35.395	-36.602	-34.393	2372.520
DE_best_1_bin	-32.592	0.671	-32.504	-34.150	-31.059	2516.240
DE_rand_1_bin	-32.704	0.499	-32.623	-34.079	-31.921	2486.190
DE_randbest_1_bin	-33.280	0.626	-33.274	-34.513	-31.781	2590.990
DE_best_2_bin	-31.581	1.033	-31.590	-34.279	-29.521	2631.400
DE_rand_2_bin	-33.217	0.538	-33.208	-35.363	-32.194	2708.290

Table 6: Function 6: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -40.070400
best DE cost (mean): -36.326800
best DE: DE_best_1_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_best_1_exp then **GA is better than DE_best_1_exp**.

5.7 Function 7: Stretch V Sine Wave

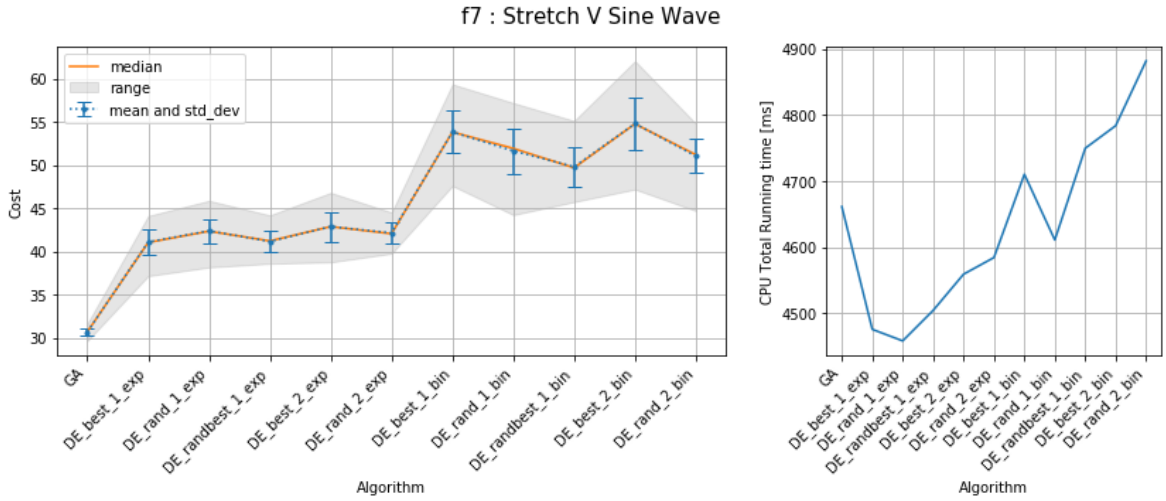

Figure 7: Cost and CPU total running time of Function 7: Stretch V Sine Wave

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	30.665	0.471	30.678	29.633	31.654	4661.080
DE_best_1_exp	41.091	1.458	41.033	37.153	44.118	4475.330
DE_rand_1_exp	42.384	1.404	42.341	38.162	45.878	4457.850
DE_randbest_1_exp	41.151	1.272	41.212	38.592	44.193	4503.920
DE_best_2_exp	42.875	1.713	42.874	38.752	46.806	4558.770
DE_rand_2_exp	42.142	1.243	42.048	39.757	44.523	4584.040
DE_best_1_bin	53.829	2.475	53.828	47.597	59.374	4710.070
DE_rand_1_bin	51.646	2.602	51.935	44.228	57.208	4610.680
DE_randbest_1_bin	49.805	2.330	49.684	45.737	55.137	4749.480
DE_best_2_bin	54.797	3.083	54.777	47.194	62.068	4783.610
DE_rand_2_bin	51.045	1.974	51.205	44.699	54.809	4881.670

Table 7: Function 7: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 30.664900
best DE cost (mean): 41.090900
best DE: DE_best_1_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_best_1_exp then **GA is better than DE_best_1_exp**.

5.8 Function 8: Ackley One

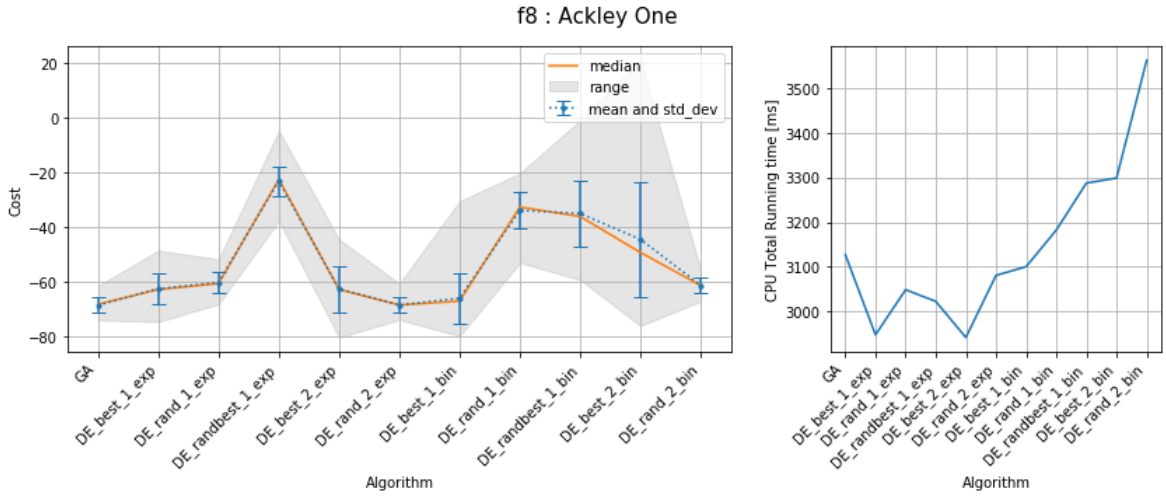

Figure 8: Cost and CPU total running time of Function 8: Ackley One

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-68.615	2.798	-68.254	-74.087	-61.435	3126.870
DE_best_1_exp	-62.618	5.764	-62.828	-74.697	-48.541	2947.120
DE_rand_1_exp	-60.235	3.611	-60.705	-68.175	-51.756	3048.230
DE_randbest_1_exp	-23.060	5.377	-22.526	-37.711	-4.706	3022.120
DE_best_2_exp	-62.723	8.563	-62.830	-80.437	-44.481	2941.100
DE_rand_2_exp	-68.522	2.789	-68.553	-73.931	-60.632	3080.420
DE_best_1_bin	-66.035	9.275	-67.084	-79.900	-30.430	3100.030
DE_rand_1_bin	-33.812	6.734	-32.641	-53.044	-20.210	3182.070
DE_randbest_1_bin	-35.009	11.953	-36.099	-59.294	-0.737	3287.650
DE_best_2_bin	-44.479	21.128	-49.283	-76.128	21.145	3298.820
DE_rand_2_bin	-61.283	2.866	-61.373	-67.186	-55.517	3563.030

Table 8: Function 8: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -68.615300
best DE cost (mean): -68.521700
best DE: DE_rand_2_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_exp
confidence interval = 95%
p value: 0.866900
Null Hypothesis is true with 95% of cofidence interval: **the cost of GA is equal to the cost of DE_rand_2_exp**.

5.9 Function 9: Ackley Two

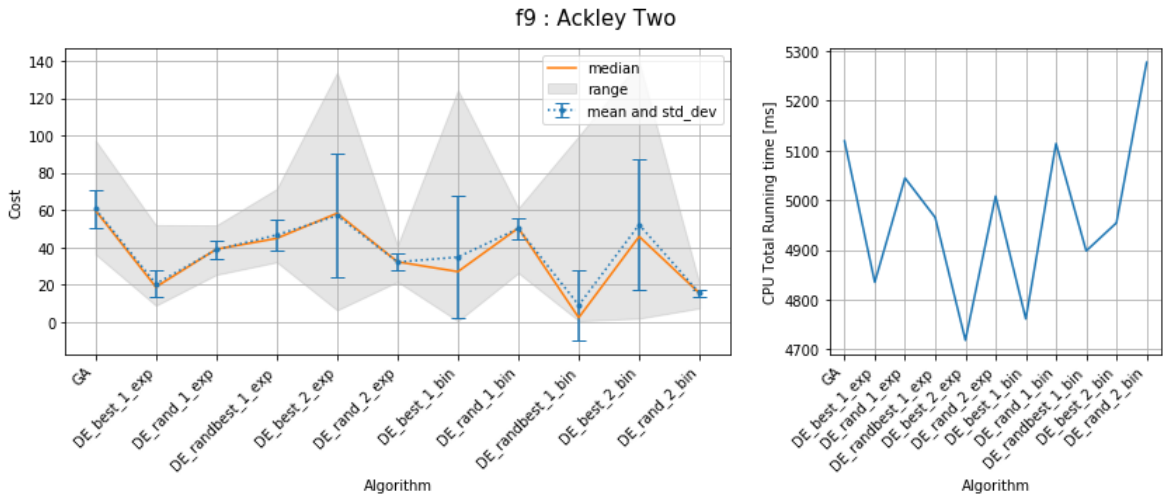

Figure 9: Cost and CPU total running time of Function 9: Ackley Two

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	60.636	10.164	59.826	36.270	97.436	5118.920
DE_best_1_exp	20.410	7.100	18.697	8.761	52.007	4834.950
DE_rand_1_exp	38.721	5.110	39.014	25.299	51.858	5044.620
DE_randbest_1_exp	46.666	8.474	44.816	32.060	71.422	4965.250
DE_best_2_exp	57.006	33.181	58.305	6.246	133.930	4718.370
DE_rand_2_exp	32.124	4.481	32.228	21.543	41.104	5008.020
DE_best_1_bin	34.759	32.776	26.968	0.386	124.512	4761.210
DE_rand_1_bin	50.078	5.938	50.376	26.227	61.097	5113.780
DE_randbest_1_bin	8.953	18.793	2.261	0.527	99.830	4897.940
DE_best_2_bin	52.178	35.027	45.816	1.823	139.311	4954.370
DE_rand_2_bin	15.336	2.131	15.316	7.345	20.440	5277.080

Table 9: Function 9: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 60.635600
best DE cost (mean): 8.952710
best DE: DE_randbest_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_randbest_1_bin
confidence interval = 95%
p value: 0.000000
DE_randbest_1_bin obtains lower cost than GA then **DE_randbest_1_bin is better than GA**.

5.10 Function 10: Egg Holder

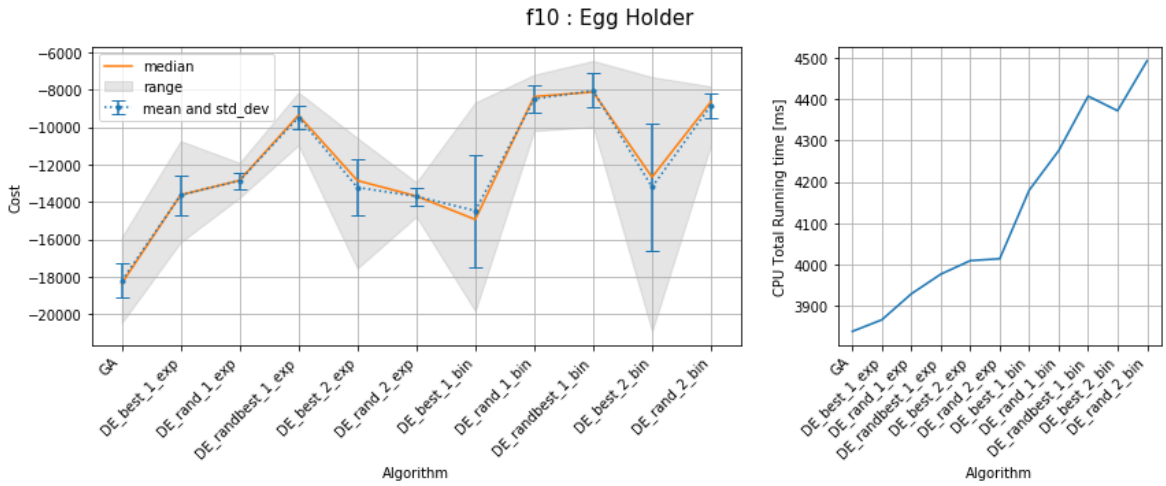

Figure 10: Cost and CPU total running time of Function 10: Egg Holder

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-18226.300	915.881	-18373.200	-20476.500	-15880.700	3839.400
DE_best_1_exp	-13635.300	1043.520	-13621.600	-16154.000	-10730.700	3867.320
DE_rand_1_exp	-12848.900	447.916	-12845.400	-13772.600	-11899.400	3929.930
DE_randbest_1_exp	-9469.860	647.241	-9353.080	-10953.300	-8129.610	3977.400
DE_best_2_exp	-13220.500	1520.260	-12851.000	-17552.400	-10572.200	4010.010
DE_rand_2_exp	-13696.800	487.713	-13682.800	-14779.800	-12941.900	4014.570
DE_best_1_bin	-14468.300	2996.410	-14935.900	-19843.400	-8668.150	4180.020
DE_rand_1_bin	-8478.360	734.300	-8352.540	-10190.100	-7180.200	4275.380
DE_randbest_1_bin	-8036.620	916.736	-8106.190	-9971.340	-6435.620	4406.880
DE_best_2_bin	-13206.800	3440.280	-12669.200	-20928.800	-7302.990	4371.700
DE_rand_2_bin	-8821.480	653.647	-8629.970	-10926.000	-7807.520	4492.260

Table 10: Function 10: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -18226.300000
best DE cost (mean): -14468.300000
best DE: DE_best_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_bin
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_best_1_bin then **GA is better than DE_best_1_bin**.

5.11 Function 11: Rana

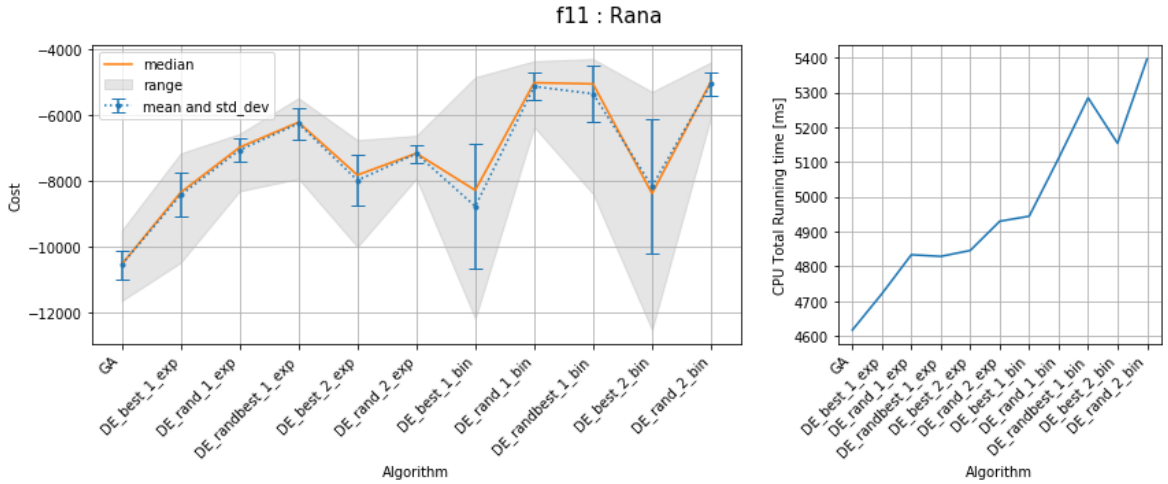

Figure 11: Cost and CPU total running time of Function 11: Rana

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-10546.200	440.372	-10547.300	-11654.400	-9505.950	4617.120
DE_best_1_exp	-8429.380	668.964	-8353.660	-10489.400	-7158.370	4721.350
DE_rand_1_exp	-7061.110	353.680	-6978.280	-8313.310	-6563.800	4833.610
DE_randbest_1_exp	-6252.450	477.028	-6213.170	-7932.240	-5474.680	4828.880
DE_best_2_exp	-7981.080	773.981	-7820.980	-10008.800	-6754.880	4845.910
DE_rand_2_exp	-7170.400	264.664	-7160.110	-7928.800	-6619.250	4929.900
DE_best_1_bin	-8775.320	1907.570	-8282.570	-12196.800	-4839.800	4944.400
DE_rand_1_bin	-5122.620	415.338	-5007.840	-6357.640	-4347.190	5111.310
DE_randbest_1_bin	-5345.630	845.420	-5044.440	-8401.070	-4283.130	5284.960
DE_best_2_bin	-8162.530	2033.390	-8385.280	-12542.300	-5285.150	5153.990
DE_rand_2_bin	-5037.860	360.314	-4972.590	-6054.450	-4384.930	5396.710

Table 11: Function 11: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -10546.200000
best DE cost (mean): -8775.320000
best DE: DE_best_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_bin
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_best_1_bin then **GA is better than DE_best_1_bin**.

5.12 Function 12: Pathological

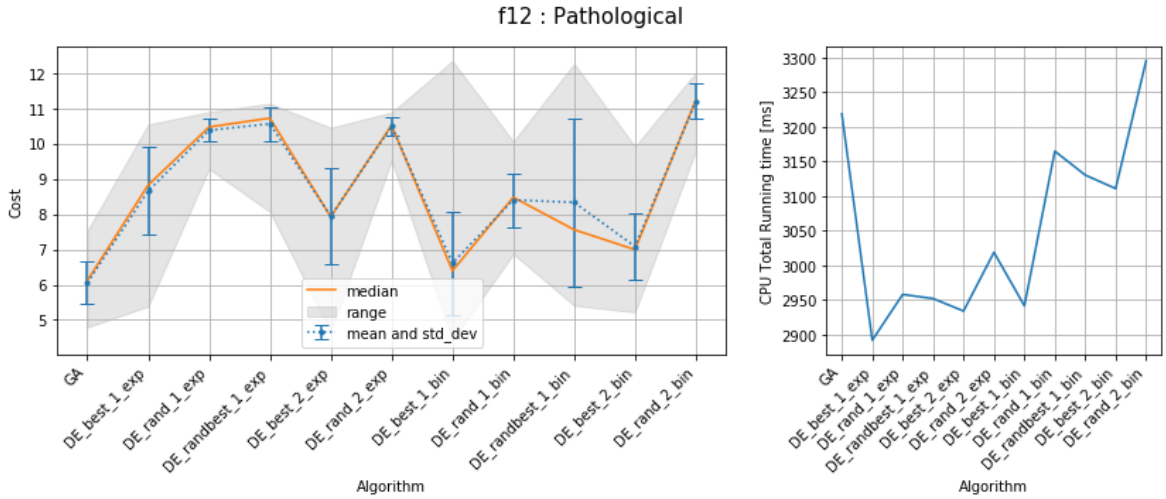

Figure 12: Cost and CPU total running time of Function 12: Pathological

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	6.054	0.602	6.113	4.790	7.535	3218.660
DE_best_1_exp	8.655	1.238	8.824	5.386	10.544	2891.730
DE_rand_1_exp	10.378	0.319	10.471	9.287	10.901	2957.840
DE_randbest_1_exp	10.562	0.474	10.723	8.065	11.137	2951.870
DE_best_2_exp	7.941	1.351	7.925	4.665	10.451	2933.840
DE_rand_2_exp	10.491	0.264	10.520	9.524	10.890	3018.680
DE_best_1_bin	6.614	1.457	6.392	4.424	12.349	2941.320
DE_rand_1_bin	8.402	0.764	8.477	6.870	10.065	3164.780
DE_randbest_1_bin	8.334	2.377	7.559	5.407	12.267	3130.240
DE_best_2_bin	7.082	0.961	6.993	5.218	9.919	3110.790
DE_rand_2_bin	11.201	0.502	11.234	9.793	12.019	3295.300

Table 12: Function 12: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 6.054130
best DE cost (mean): 6.613880
best DE: DE_best_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_1_bin
confidence interval = 95%
p value: 0.012000
GA obtains lower cost than DE_best_1_bin then **GA is better than DE_best_1_bin**.

5.13 Function 13: Michalewicz

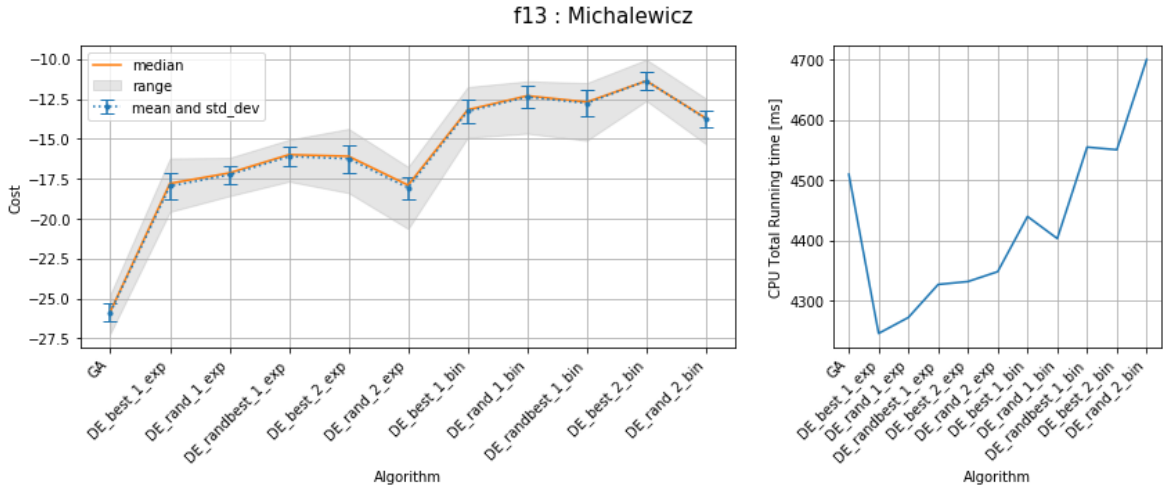

Figure 13: Cost and CPU total running time of Function 13: Michalewicz

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-25.887	0.552	-25.835	-27.196	-24.732	4509.420
DE_best_1_exp	-17.967	0.800	-17.802	-19.574	-16.230	4245.820
DE_rand_1_exp	-17.257	0.541	-17.142	-18.587	-16.191	4272.270
DE_randbest_1_exp	-16.114	0.600	-15.996	-17.676	-15.058	4326.960
DE_best_2_exp	-16.249	0.852	-16.099	-18.393	-14.376	4331.730
DE_rand_2_exp	-18.072	0.679	-17.908	-20.645	-16.758	4348.170
DE_best_1_bin	-13.277	0.708	-13.189	-14.930	-11.754	4439.490
DE_rand_1_bin	-12.388	0.695	-12.308	-14.673	-11.395	4403.000
DE_randbest_1_bin	-12.776	0.807	-12.700	-15.115	-11.504	4554.450
DE_best_2_bin	-11.374	0.601	-11.376	-12.625	-10.038	4550.100
DE_rand_2_bin	-13.749	0.532	-13.717	-15.330	-12.476	4699.510

Table 13: Function 13: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -25.887200
best DE cost (mean): -18.071900
best DE: DE_rand_2_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_rand_2_exp then **GA is better than DE_rand_2_exp**.

5.14 Function 14: Masters' Cosine Wave

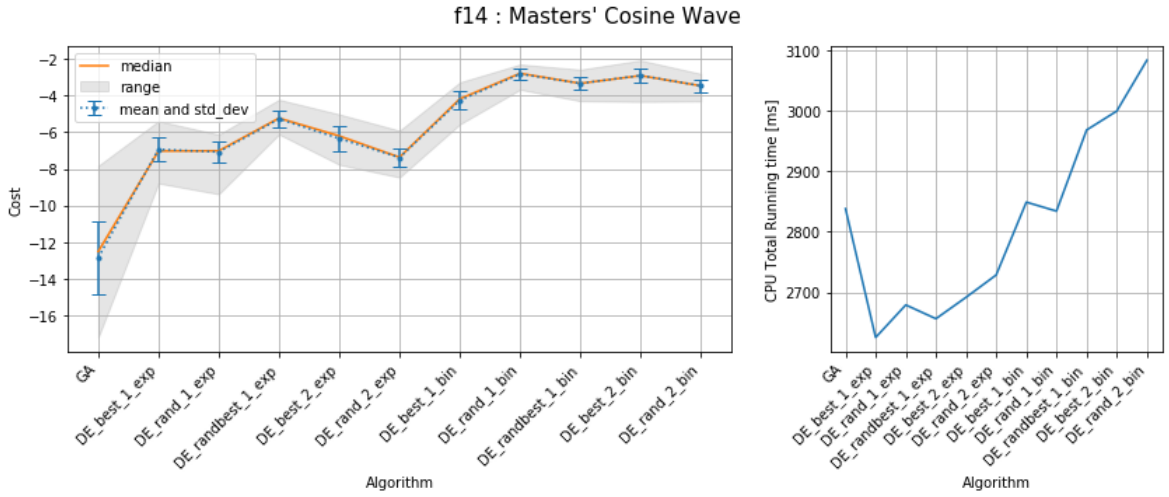

Figure 14: Cost and CPU total running time of Function 14: Masters' Cosine Wave

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	-12.842	1.966	-12.510	-17.187	-7.832	2838.080
DE_best_1_exp	-6.943	0.625	-7.034	-8.789	-5.398	2625.200
DE_rand_1_exp	-7.084	0.589	-7.023	-9.378	-6.132	2679.030
DE_randbest_1_exp	-5.286	0.448	-5.241	-6.123	-4.230	2656.230
DE_best_2_exp	-6.350	0.684	-6.212	-7.774	-5.031	2691.650
DE_rand_2_exp	-7.389	0.519	-7.371	-8.466	-5.919	2728.580
DE_best_1_bin	-4.274	0.513	-4.195	-5.576	-3.289	2849.020
DE_rand_1_bin	-2.836	0.312	-2.804	-3.683	-2.312	2834.160
DE_randbest_1_bin	-3.351	0.360	-3.349	-4.309	-2.595	2968.260
DE_best_2_bin	-2.921	0.391	-2.926	-4.347	-2.092	2999.520
DE_rand_2_bin	-3.483	0.330	-3.484	-4.323	-2.823	3083.540

Table 14: Function 14: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): -12.841700
best DE cost (mean): -7.389280
best DE: DE_rand_2_exp
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_exp
confidence interval = 95%
p value: 0.000000
GA obtains lower cost than DE_rand_2_exp then **GA is better than DE_rand_2_exp**.

5.15 Function 15: Quartic

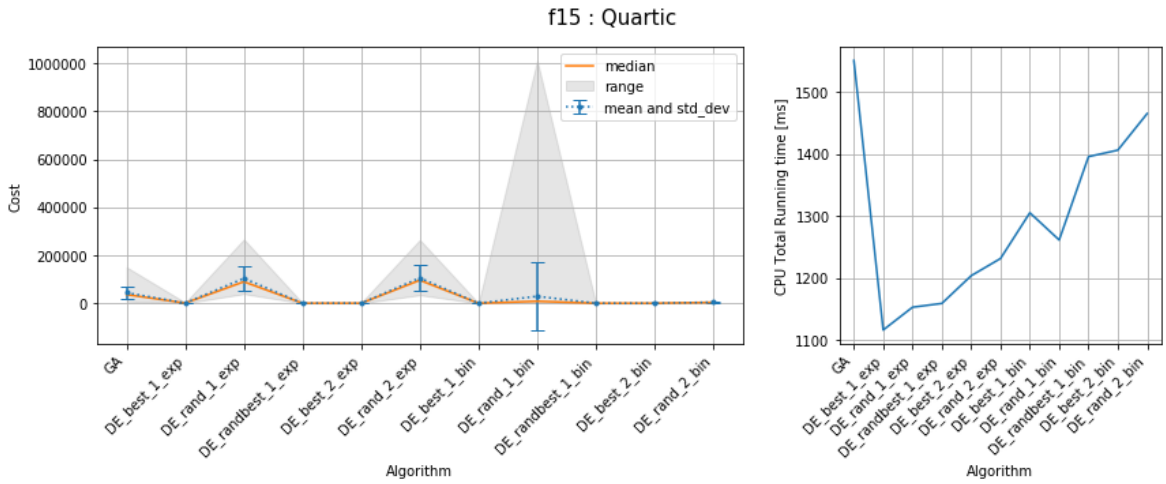

Figure 15: Cost and CPU total running time of Function 15: Quartic

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	43658.600	28045.300	35983.300	7846.720	150243.000	1550.680
DE_best_1_exp	806.342	719.590	593.730	55.636	2999.920	1116.150
DE_rand_1_exp	103284.000	49304.900	89445.200	38826.800	266325.000	1152.880
DE_randbest_1_exp	73.893	145.668	33.921	3.337	955.929	1158.990
DE_best_2_exp	253.454	504.449	48.181	0.010	2479.520	1203.700
DE_rand_2_exp	105577.000	55484.600	95674.200	33958.200	264131.000	1231.430
DE_best_1_bin	0.105	0.380	0.004	0.000	2.426	1305.180
DE_rand_1_bin	28785.100	140387.000	8082.490	1888.710	1010990.000	1261.300
DE_randbest_1_bin	3.999	16.630	0.027	0.000	114.900	1395.720
DE_best_2_bin	0.016	0.029	0.005	0.000	0.164	1406.170
DE_rand_2_bin	3825.140	1627.090	3518.840	1554.950	8387.470	1465.150

Table 15: Function 15: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 43658.600000
best DE cost (mean): 0.016239
best DE: DE_best_2_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_best_2_bin
confidence interval = 95%
p value: 0.000000
DE_best_2_bin obtains lower cost than GA then **DE_best_2_bin is better than GA**.

5.16 Function 16: Levy

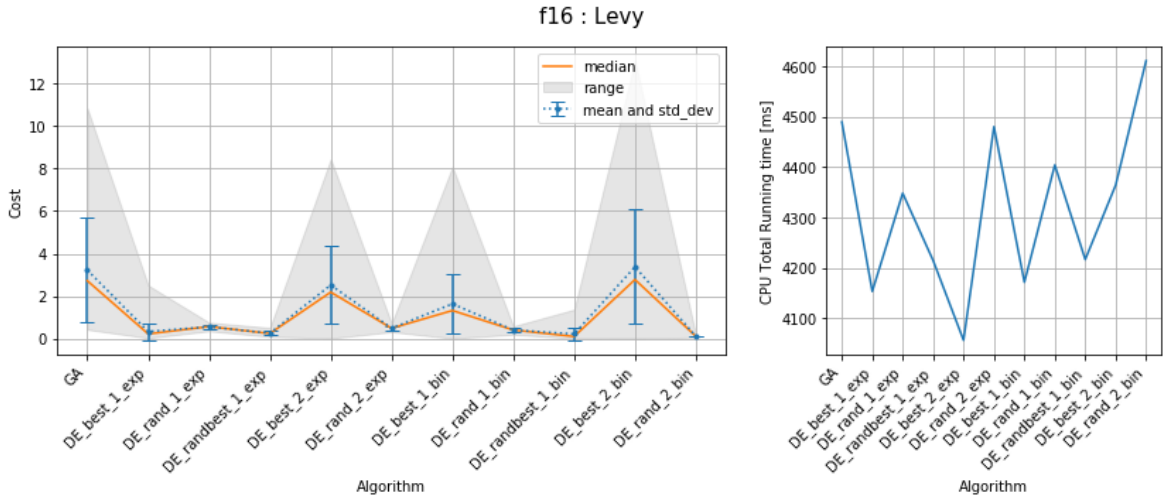

Figure 16: Cost and CPU total running time of Function 16: Levy

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	3.220	2.454	2.718	0.417	10.843	4489.610
DE_best_1_exp	0.324	0.404	0.212	0.020	2.503	4153.580
DE_rand_1_exp	0.548	0.097	0.558	0.347	0.752	4348.050
DE_randbest_1_exp	0.245	0.103	0.231	0.082	0.505	4214.800
DE_best_2_exp	2.517	1.845	2.181	0.004	8.449	4056.670
DE_rand_2_exp	0.481	0.105	0.478	0.306	0.707	4479.960
DE_best_1_bin	1.629	1.408	1.315	0.000	8.091	4171.730
DE_rand_1_bin	0.397	0.087	0.398	0.184	0.575	4404.010
DE_randbest_1_bin	0.208	0.292	0.090	0.000	1.363	4216.740
DE_best_2_bin	3.374	2.696	2.773	0.006	13.085	4364.160
DE_rand_2_bin	0.106	0.029	0.100	0.048	0.165	4610.660

Table 16: Function 16: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 3.220140
best DE cost (mean): 0.105905
best DE: DE_rand_2_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_bin
confidence interval = 95%
p value: 0.000000
DE_rand_2_bin obtains lower cost than GA then **DE_rand_2_bin is better than GA**.

5.17 Function 17: Step

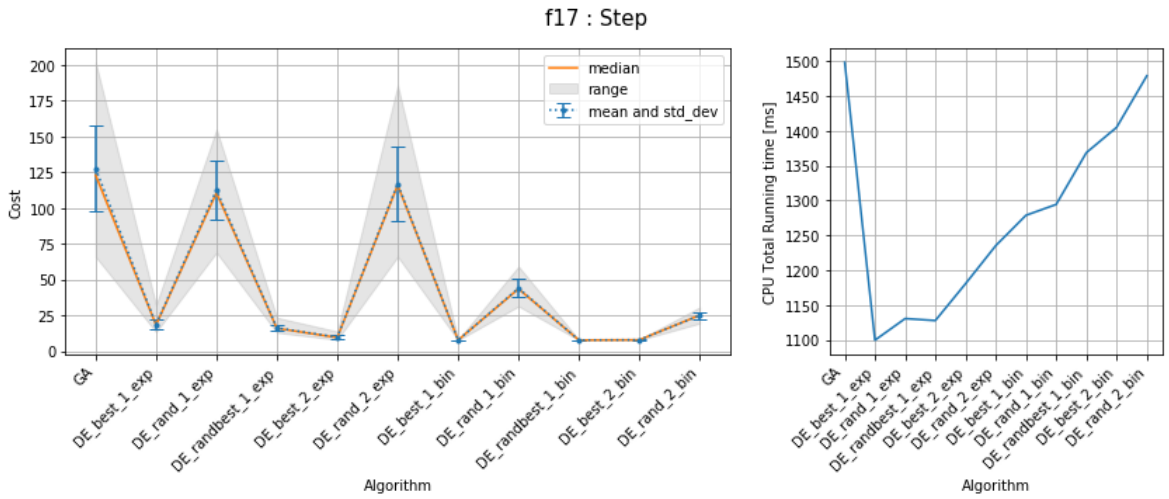

Figure 17: Cost and CPU total running time of Function 17: Step

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	127.735	29.752	123.041	65.873	201.796	1498.120
DE_best_1_exp	18.614	3.455	18.223	13.358	32.191	1099.640
DE_rand_1_exp	112.570	20.324	110.635	68.843	155.092	1130.610
DE_randbest_1_exp	16.238	2.123	15.909	12.909	23.394	1127.790
DE_best_2_exp	9.695	1.278	9.425	7.967	13.764	1180.710
DE_rand_2_exp	116.961	26.495	116.016	65.986	185.861	1235.260
DE_best_1_bin	7.717	0.067	7.706	7.624	7.971	1278.790
DE_rand_1_bin	43.947	6.483	43.452	31.336	59.197	1294.370
DE_randbest_1_bin	7.674	0.057	7.658	7.612	7.889	1368.700
DE_best_2_bin	7.836	0.094	7.815	7.667	8.108	1404.950
DE_rand_2_bin	24.961	2.397	25.062	19.401	30.667	1478.870

Table 17: Function 17: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 127.735000
best DE cost (mean): 7.673640
best DE: DE_randbest_1_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_randbest_1_bin
confidence interval = 95%
p value: 0.000000
DE_randbest_1_bin obtains lower cost than GA then **DE_randbest_1_bin is better than GA**.

5.18 Function 18: Alphine

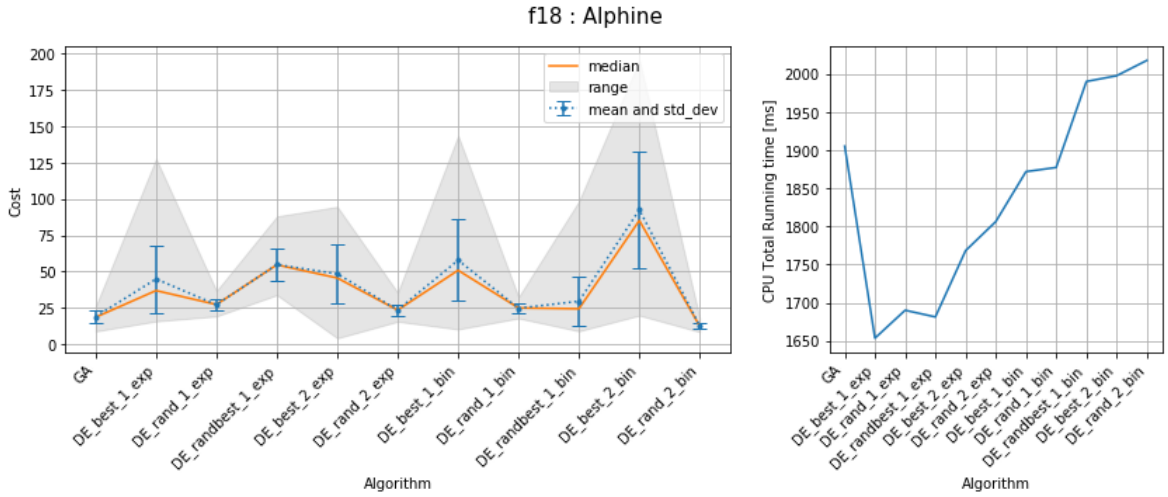

Figure 18: Cost and CPU total running time of Function 18: Alphine

algorithm	mean	std_dev	median	range_min	range_max	time_ms
GA	18.903	4.296	18.499	8.833	26.103	1905.180
DE_best_1_exp	44.836	23.021	36.868	15.752	127.641	1653.390
DE_rand_1_exp	27.128	3.951	27.255	19.172	36.741	1690.010
DE_randbest_1_exp	54.767	11.171	54.614	33.832	88.008	1681.220
DE_best_2_exp	48.362	20.063	45.537	4.092	94.633	1768.150
DE_rand_2_exp	23.537	3.785	23.411	15.557	35.538	1806.250
DE_best_1_bin	57.801	28.004	50.831	10.234	143.547	1871.850
DE_rand_1_bin	24.683	3.349	24.913	17.813	32.461	1877.280
DE_randbest_1_bin	29.555	17.094	24.332	8.935	98.477	1990.090
DE_best_2_bin	92.449	40.111	85.133	19.770	195.347	1997.580
DE_rand_2_bin	12.652	1.917	12.325	8.391	17.788	2017.690

Table 18: Function 18: Statistical Analysis of the Cost obtained by GA and DE

Two-Sample Z-Test Hypothesis Testing:
GA cost (mean): 18.902700
best DE cost (mean): 12.652100
best DE: DE_rand_2_bin
Null Hypothesis: The cost value obtained by GA is equal to the DE_rand_2_bin
confidence interval = 95%
p value: 0.000000
DE_rand_2_bin obtains lower cost than GA then **DE_rand_2_bin is better than GA**.

5.19 Summary

function_id	GA	DE_best_1_exp	DE_rand_1_exp	DE_randbest_1_exp	DE_best_2_exp	DE_rand_2_exp	DE_best_1_bin	DE_rand_1_bin	DE_randbest_1_bin	DE_best_2_bin	DE_rand_2_bin
1	1330.170	2554.650	3221.480	6393.850	3393.730	1859.860	2012.120	6833.980	6764.300	2187.930	6168.990
2	85.044	4.529	74.558	2.406	0.459	79.866	0.004	18.665	0.002	0.007	8.319
3	358923.000	8366.080	616375.000	1634.280	3820.620	753439.000	520.354	72972.600	562.328	1048.130	41223.200
4	-67715.300	-48389.100	-40661.400	-42764.300	-49615.600	-49116.300	-37136.400	-29531.800	-34185.400	-19621.900	-40013.100
5	1.603	0.993	1.474	0.956	0.217	1.475	0.015	1.121	0.011	0.023	1.048
6	-40.070	-36.327	-35.602	-35.918	-35.868	-35.457	-32.592	-32.704	-33.280	-31.581	-33.217
7	30.665	41.091	42.384	41.151	42.875	42.142	53.829	51.646	49.805	54.797	51.045
8	-68.615	-62.618	-60.235	-23.060	-62.723	-68.522	-66.035	-33.812	-35.009	-44.479	-61.283
9	60.636	20.410	38.721	46.666	57.006	32.124	34.759	50.078	8.953	52.178	15.336
10	-18226.300	-13635.300	-12848.900	-9469.860	-13220.500	-13696.800	-14468.300	-8478.360	-8036.620	-13206.800	-8821.480
11	-10546.200	-8429.380	-7061.110	-6252.450	-7981.080	-7170.400	-8775.320	-5122.620	-5345.630	-8162.530	-5037.860
12	6.054	8.655	10.378	10.562	7.941	10.491	6.614	8.402	8.334	7.082	11.201
13	-25.887	-17.967	-17.257	-16.114	-16.249	-18.072	-13.277	-12.388	-12.776	-11.374	-13.749
14	-12.842	-6.943	-7.084	-5.286	-6.350	-7.389	-4.274	-2.836	-3.351	-2.921	-3.483
15	43658.600	806.342	103284.000	73.893	253.454	105577.000	0.105	28785.100	3.999	0.016	3825.140
16	3.220	0.324	0.548	0.245	2.517	0.481	1.629	0.397	0.208	3.374	0.106
17	127.735	18.614	112.570	16.238	9.695	116.961	7.717	43.947	7.674	7.836	24.961
18	18.903	44.836	27.128	54.767	48.362	23.537	57.801	24.683	29.555	92.449	12.652

Table 19: Summary: mean cost of each optimization algorithm with different bench-mark functions, the best cost of each function is highlighted.

The table 19 shows a summary of the mean cost obtained by each algorithm, the lowest cost of each function is highlighted in yellow. It can be observed that Genetic Algorithm found the minimum values in 10 functions, in the some way Differential Algorithms are pretty good, they found the lowest cost in 9 functions. In function 8 both algorithm found the lowest cost.

6 Discussion

The previous results show that both GA and DE are good, almost half of the lowest cost are found by GA and another half by DE. Regarding to the running time it can be noticed that in general the GA are slower than DE since in GA it is necessary to do two sorting in each iteration that makes it more time consuming. Additionally the figures show that in general binomial crossover version of DE are slower than the exponential crossover.

7 Conclusion

In conclusion,

We assume that tuning manually it has achieve the best parameters

References

[1] Classes with pointer data members. `http://pages.cs.wisc.edu/~hasti/cs368/CppTutorial/NOTES/CLASSES-PTRS.html`. Accessed: 2019-04-08.

[2] Measuring cpi time in c. `https://stackoverflow.com/questions/20167685/measuring-cpu-time-in-c`. Accessed: 2020-02-11.

ANNEX

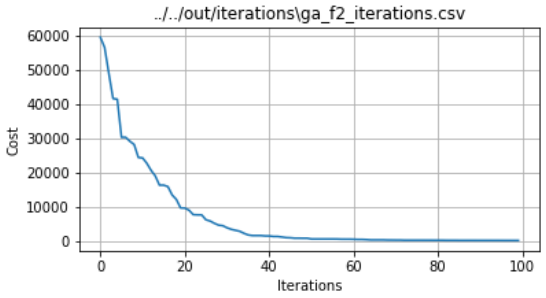

Figure 19: Genetic Algorithm: Cost vs Iterations, function2

The figure 19 shows how Genetic Algorithm Converge through different generations (iterations).

The figure 20 shows how Differential Evolution Algorithm Converge through different generations (iterations).

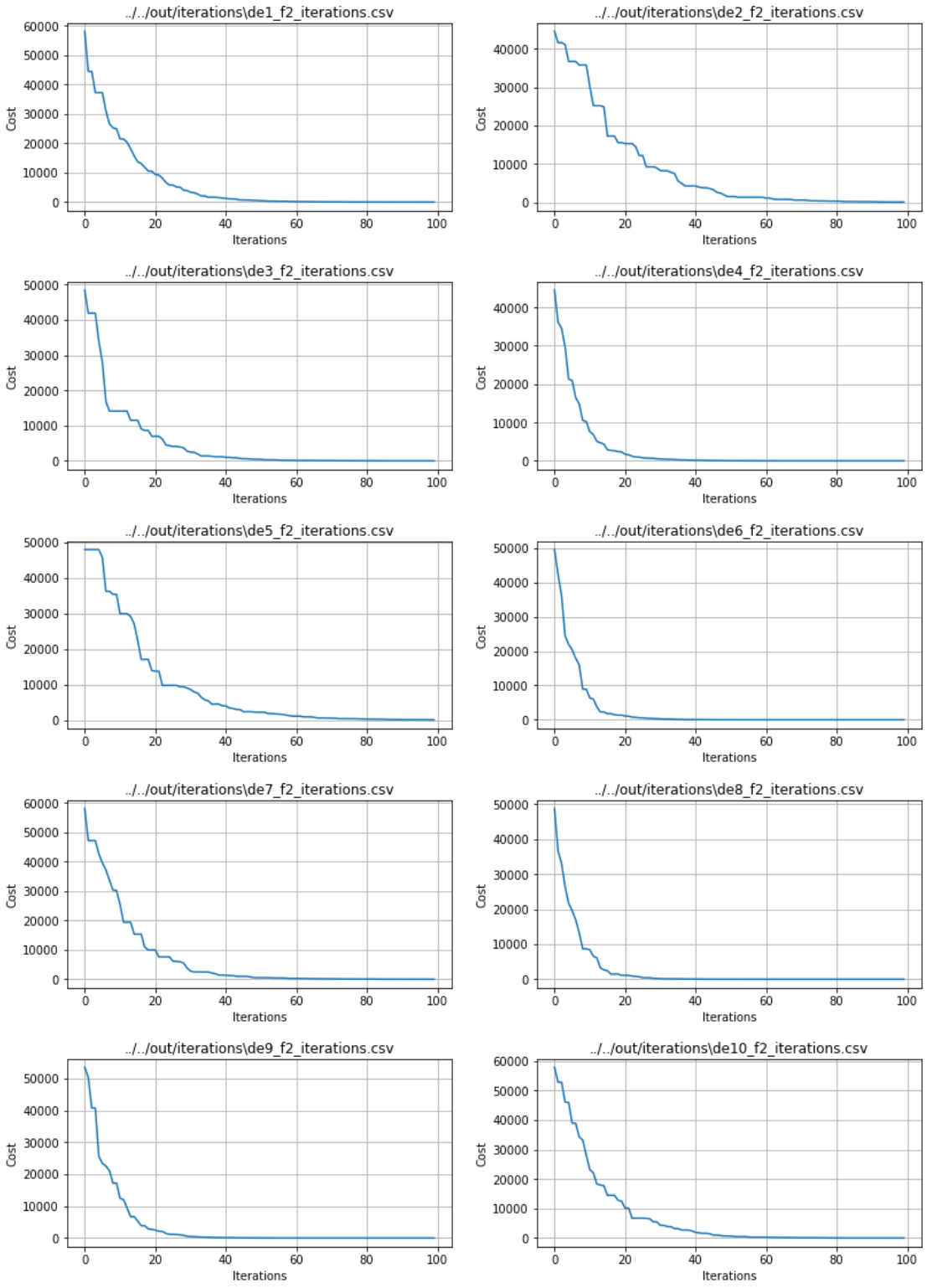

Figure 20: Differential Evolution: Cost vs Iterations, function2