CS471 Project4

JuneYeob Lee(2462 9603)

May 20, 2019

1 INTRODUCTION

Three meta-heuristics algorithms are implemented for this project. The first meta-heuristic algorithm is Particle Swarm Optimization (PSO). PSO was invented by Russell Eberhart and James Kennedy in 1995. The basic idea of PSO is updating particles until the fitness solution is close to the optimal value. The way to update particles is to calculate the velocity for each particle. The initial velocity the same as the fitness of a vector of particles in the population. Velocity is changed by using three constant parameters which are c1, c2, and k. The value for k controls the speed of how the particles are moving. Particles are updated by added the velocity to the previous particles. For the experimentation c1 was set to 0.1, c2 was set to 0,6 and k was set to 0.001. The second algorithm is Firefly Algorithm (FA) which was invented by Yang. Yang was inspired by the social behavior of fireflies. The brighter firefly has a smaller function value. Alpha, Beta, and Gamma have been set to 0.5, 0.2, and 0.1. The third algorithm is Harmony Search (HS). HS creates a new harmony vector and if that new vector has a better fitness solution than the worst vector in the population then the new vector replaces the worst vector. The bw value for HS has been set to 0.2. All three algorithms have been implemented with a population size of 500, 30 dimensions, and 500 iterations.

2 ANALYSIS

According to Figure 4.1, 4.2, and 4.3, PSO gave the best solutions with 500 generations. PSO also has the fastest execution time among the three algorithms. The global best value for

the DeJong function only improved for HS. The worst values continued to decrease by each iteration. HS algorithm has the fastest execution time for the DeJong function. All three algorithms did not work well with the Rosenbrock function. PSO did not give an improved solution. Even though the solution has improved with FA and HS algorithms, itâĂŹs too far from the target value. RastrignâĂŹs function only works with HS algorithm. However, the last global best value is still far from the target value. PSO shows the best performance with Griewangk, Sine Envelope,and Stretched V function compared to FA and HS. HS gave the lowest global minimum value for Ackley1 Function. PSO showed a good performance for Ackley2 function too. On the other hand, HS and FA algorithms shows small changes for the global best value. For the EggHolder function, PSO had a big improvement. The ïňĄnal minimum value is similar to FA and HSâĂŹ ïňĄnal minimum value. Solution for the Pathological function does not improve that much with PSO, FA, and HS because the initial solution is already close to the target value. Only HS algorithm improved solutions for Quartic function, Levy function and Step function. Overall, the functions with PSO has dynamic change however, the ïňĄnal global minimum values with 3 algorithms are similar. PSO has failed to improve solution of DeJong, Rosenbrock, Rastrigin, Stretched V, Quartic, Levy, and Step Functions. All the functions that PSO has failed have value which are too big or too close to the target value. FA has failed to improve solution for most of functions but the worst value decreases by each iteration. HS has failed to improve solutions of Sine Envelop, Stretched V, EggHolder, Rana, Pathological, and MastersâĂŹ. HS succeed to work on functions that PSO failed at. The execution time is the shortest for PSO, then HS, and finally FA. Stagnation happens to the generation when individuals are converging together. Speed of stagnating happens in the order of FA, HS, and PSO because the slope of the graph is the steepest on the FA stagnation graph.

3 Conclusion

According to the results obtained, the three new algorithms are performing better than the previous algorithm from the previous report. Compared to Differential Evolution algorithm, PSO , FA, and HS have better results. These results are improved because all three algorithms are able to create new vectors for the next population and replace the worst from the current population. DE is more dependent on luck which results in worse results. From the experiment, the constant values c1, c2, and k are important values that need to be continuously tuned to see improvements in the fitness results. To avoid stagnation k should be as small as possible. However, the best fitness will not be improved if k is too small. More experimentation is required for each function since the constant values have a large range that could be chosen from. The functions which are not optimized need more experimentations and changing of the constants to see if there are any improvements. Another experimentation that could be had is using negative values for the constants and seeing how that effects the results.

Table 3.1: 18 Functions with PSO (c1=0.1 c2 = 0.6 ,and k =0.001)

	AVERAGE	SD	MIN	MAX	MED	STAGNATION	Time(milisec)
F(x0)	5093.854787	3991.487934	3442.602522	9087.418892	6265.010707	465	1018
F(x1)	41810.29555	0	41810.29555	41810.29555	41810.29555	0	412
F(x2)	20200000000	0	20200000000	20200000000	20200000000	0	381
F(x3)	1368424.865	0	1368424.865	1368424.865	1368424.865	0	753
F(x4)	293.3514875	62.58879206	247.822322	336.3362406	292.0792813	not found	1013
F(x5)	-31.11884396	3.836626472	-32.33804505	-26.91223586	-29.62514046	0	1174
F(x6)	53.40265856	0	53.40265856	53.40265856	53.40265856	17	2089
F(x7)	358.6237818	22.50441067	358.065289	389.8913318	373.9783104	0	1396
F(x8)	466.9875754	39.60523275	461.0521031	517.0623604	489.0572318	79	2373
F(x9)	-5590.979902	156.1566342	-5598.046943	-5377.208113	-5487.627528	21	1426
F(x10)	-8381.886007	7574.679221	-13496.3363	-2784.12222	-8140.229262	425	1896
F(x11)	6.09367351	2.35317587	5.715666627	9.043559857	7.379613242	402	1205
F(x12)	-8.843736255	1.13307806	-8.867135367	-7.264721008	-8.065928188	0	3227
F(x13)	-14.08531766	3.18353656	-14.94857321	-10.44637263	-12.69747292	121	1763
F(x14)	2044367309	0	2044367309	2044367309	2044367309	0	1020
F(x15)	166.2637044	0	166.2637044	166.2637044	166.2637044	0	1161
F(x16)	61576.33613	0	61576.33613	61576.33613	61576.33613	0	370
F(x17)	462.6516771	99.58420595	442.8349819	583.6683165	513.2516492	279	903

Table 3.3: 18 Functions with FA(Alpha =0.5 , Beta =0.2, and gamma =1.0)

FA Algorithm	Average	SD	MIN	MAX	Median	Stagnation	Time(milisec)
F(X0)	9660.620892	86.0249995	9583.003266	9704.660987	9643.832127	not found	270890
F(X1)	54735.82697	0	54735.82697	54735.82697	54735.82697	not found	121350
F(X2)	17304000000	2474873734	16100000000	19600000000	17850000000	not found	104283
F(X3)	1335026.692	0	1335026.692	1335026.692	1335026.692	not found	237667
F(X4)	376.9270359	0	376.9270359	376.9270359	376.9270359	not found	245076
F(X5)	-23.85513386	0	-23.8548763	-23.8548763	-23.8548763	19	623816
F(X6)	53.40265856	0	53.40265856	53.40265856	53.40265856	0	1795
F(X7)	400.9608658	0	400.9608658	400.9608658	400.9608658	93	574829
F(X8)	517.6642775	0	517.6833195	517.6833195	517.6833195	17	672221
F(X9)	-4796.459627	241.0068338	-4797.758781	-4456.923648	-4627.341215	22	674274
F(X10)	-3050.518165	108.1287051	-3053.181848	-2900.264767	-2976.723308	17	1112978
F(X11)	9.508747655	0	9.508747655	9.508747655	9.508747655	22	668903
F(X12)	-8.673376677	0	-8.673376677	-8.673376677	-8.673376677	36	692876
F(X13)	-11.10909866	0	-11.10909866	-11.10909866	-11.10909866	34	463965
F(X14)	2700000000	0	2700000000	2700000000	2700000000	16	294446
F(X15)	233.9324476	3.951532886	236.6309362	242.2192476	239.4250919	not found	364276
F(X16)	55788.41626	0	55828.27152	55828.27152	55828.27152	26	131132
F(X17)	529.8729736	0	529.8729736	529.8729736	529.8729736	58	262550

Table 3.4: 18 Functions with HS (HCMR =0.7,PAR = 0.3 , and bw =0.2)

	Average	SD	MAX	MIN	MED	Stagnation	Time(milisec)
F(x0)	8844.469152	458.8738528	9216.351439	8567.405813	8891.878626	not found	1580
F(x1)	51452.21337	9241.441433	58075.12568	45005.75387	51540.43978	not found	402
F(x2)	15625400000	1555634919	17700000000	15500000000	16600000000	not found	366
F(x3)	1248513.72	234999.833	1520632.055	1188292.104	1354462.08	not found	1662
F(x4)	302.3976897	18.53438547	321.9515279	295.7399486	308.8457383	not found	1744
F(x5)	-24.99161234	0	-24.99161234	-24.99161234	-24.99161234	not found	2031
F(x6)	53.40265856	0	53.40265856	53.40265856	53.40265856	not found	6408
F(x7)	345.616158	60.62787572	404.8592717	319.1185076	361.9888897	not found	2418
F(x8)	492.5104039	18.06879716	510.0056948	484.4525568	497.2291258	not found	6130
F(x9)	-5346.952269	0	-5346.952269	-5346.952269	-5346.952269	not found	2566
f(x10)	-4345.911303	0	-4345.911303	-4345.911303	-4345.911303	not found	4540
F(x11)	9.217491288	0	9.217491288	9.217491288	9.217491288	not found	1978
F(x12)	-8.032160861	0.9621151	-6.989913514	-8.350549737	-7.670231626	not found	4685
F(x13)	-11.08516436	0	-11.08516436	-11.08516436	-11.08516436	not found	3719
F(x14)	2266420000	403050865.3	2630000000	2060000000	2345000000	not found	2088
F(x15)	170.3477561	46.68213052	208.9443593	140.9630811	175.9351083	not found	2397
F(x16)	47960.69274	10283.70074	56111.43004	41568.08098	48839.75551	not found	394
F(x17)	516.4072239	20.2419601	532.2840056	503.6575511	517.9707784	not found	1589

Function0.pdf

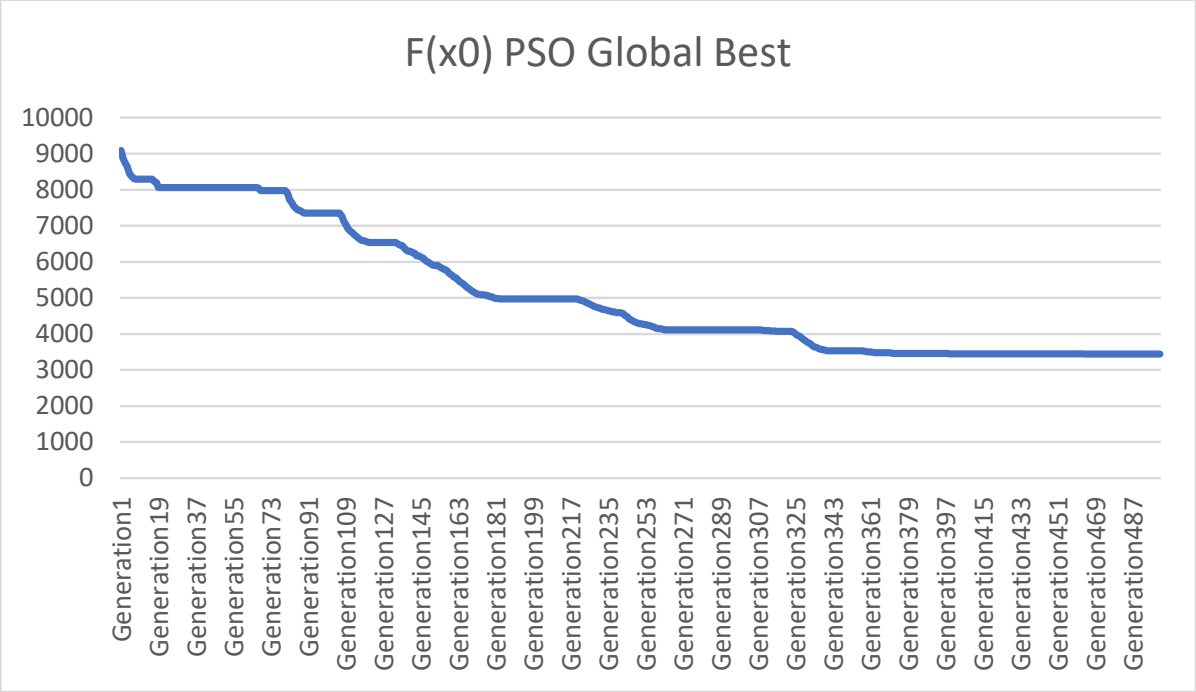


Figure 3.1: PSO Algorithm with Schwefel Function

Function0.pdf

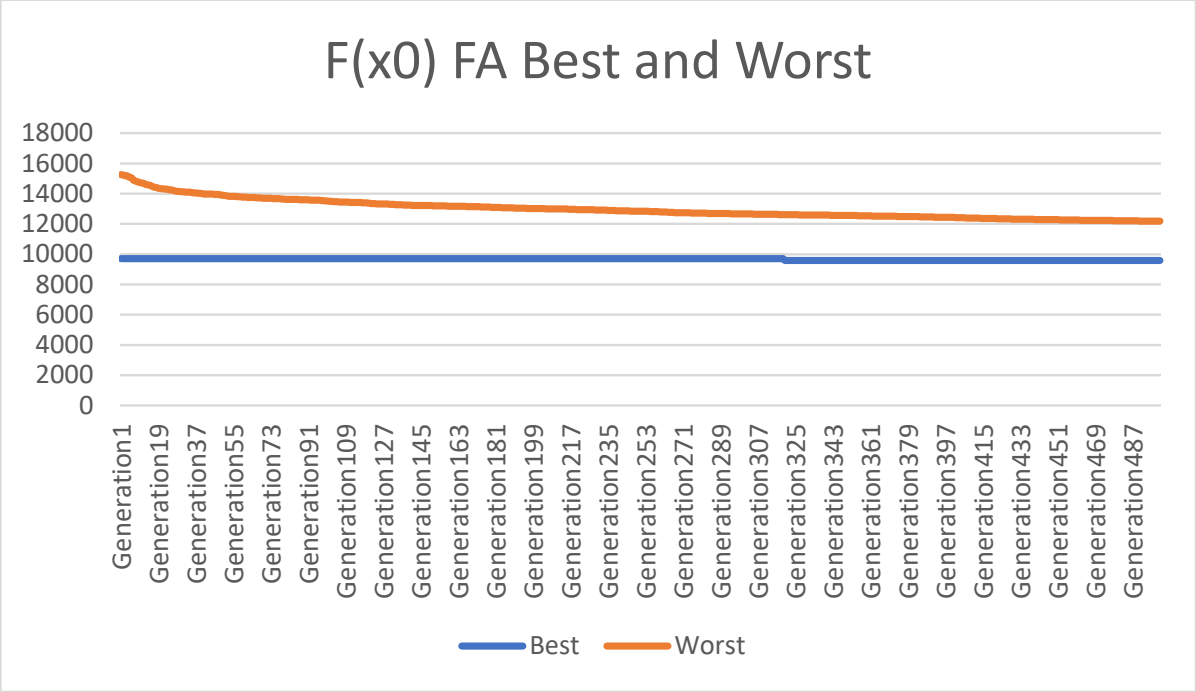


Figure 3.2: FA Algorithm with Schwefel Function

Function0.pdf

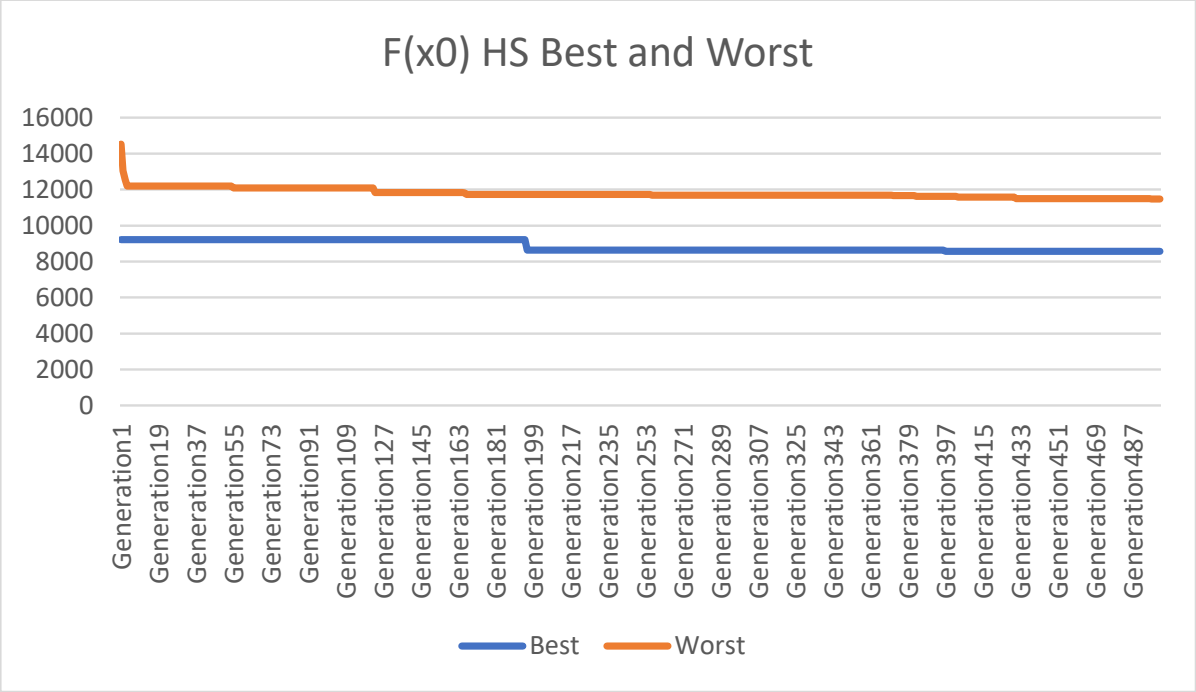


Figure 3.3: HS Algorithm with Schwefel Function

Function1.pdf

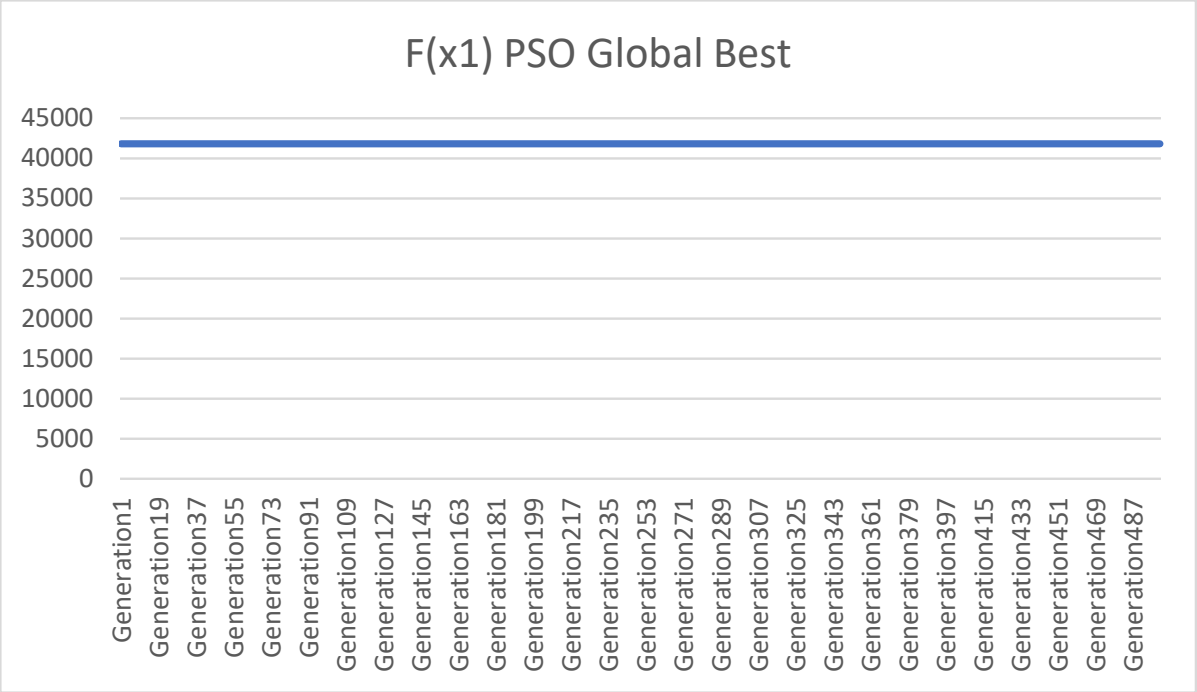


Figure 3.4: PSO Algorithm with De Jong Function

Function1.pdf

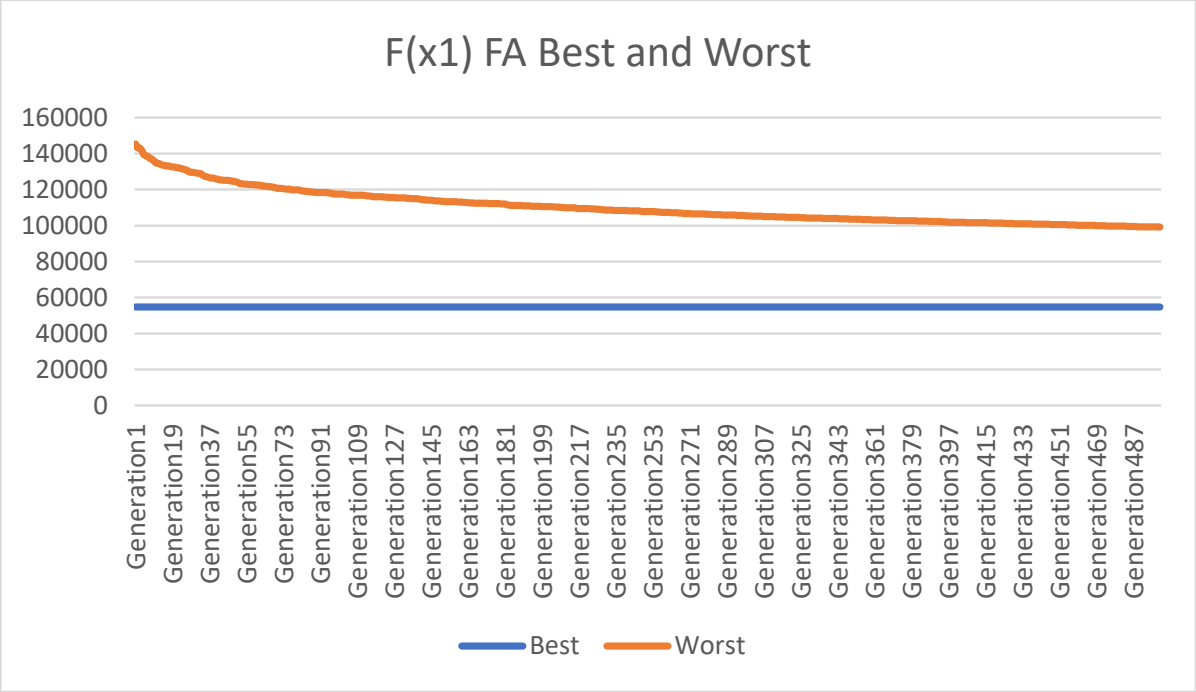


Figure 3.5: FA Algorithm with De Jong Function

Function1.pdf

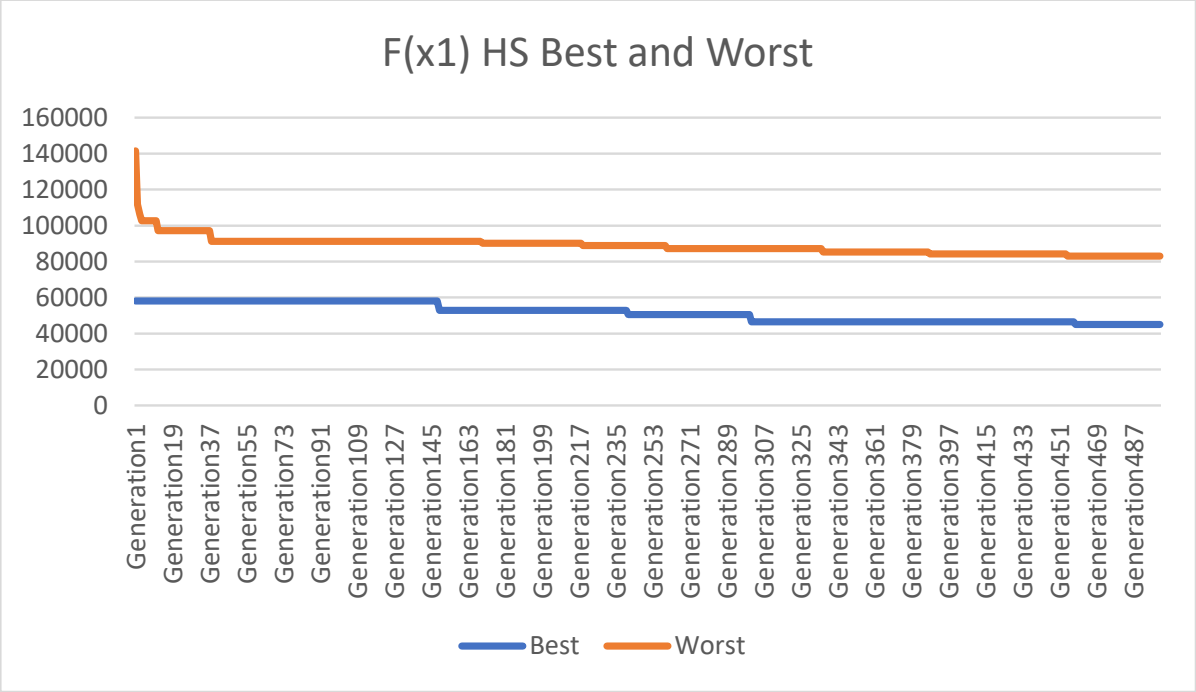


Figure 3.6: HS Algorithm with De Jong Function

Function2.pdf

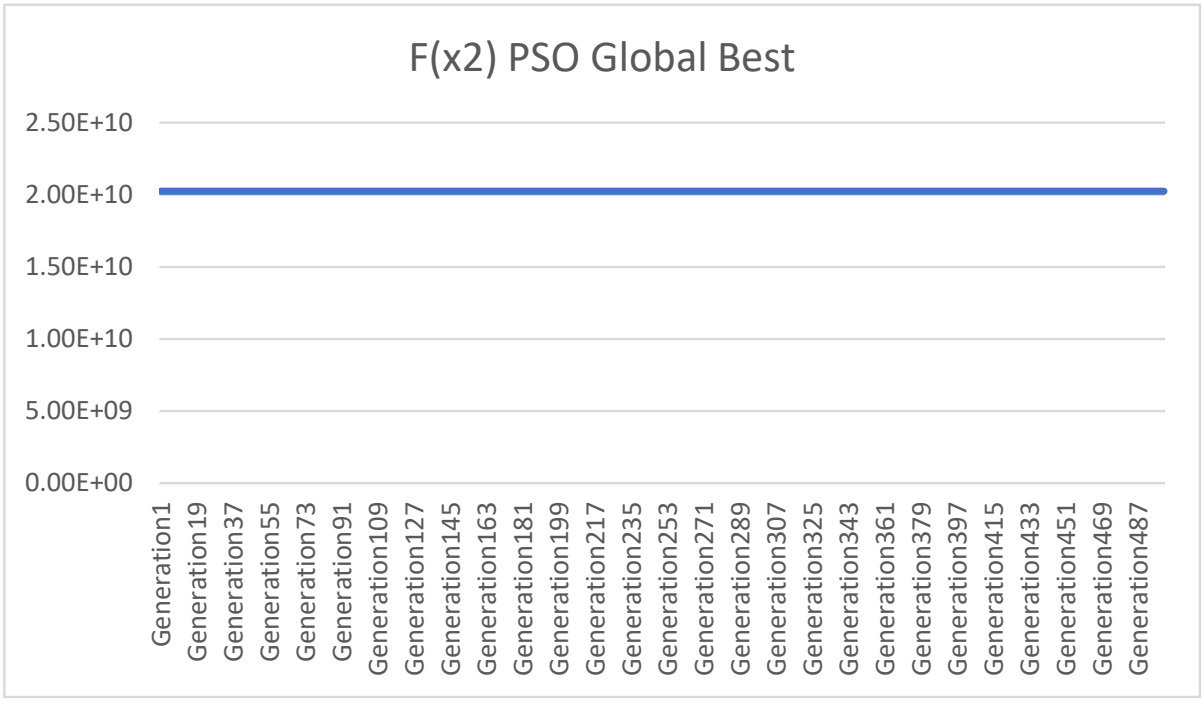


Figure 3.7: PSO Algorithm with Rosenbrock Function

Function2.pdf

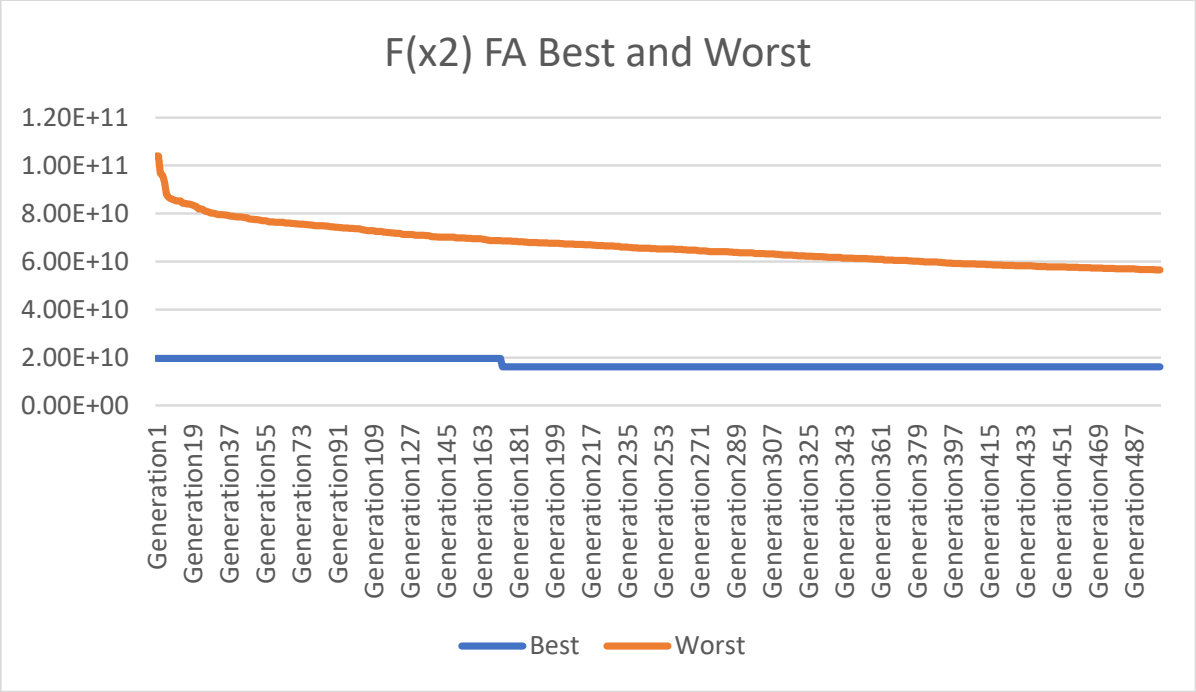


Figure 3.8: FA Algorithm with Rosenbrock Function

Function2.pdf

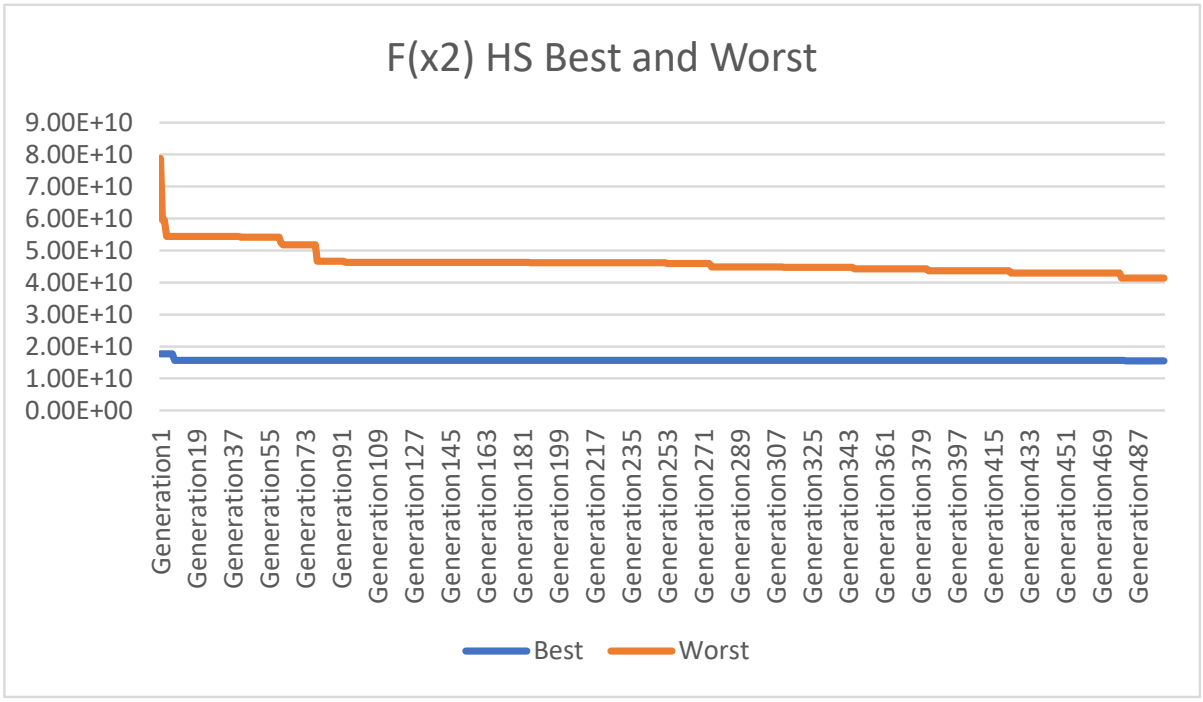


Figure 3.9: HS Algorithm with Rosenbrock Function

Function3.pdf

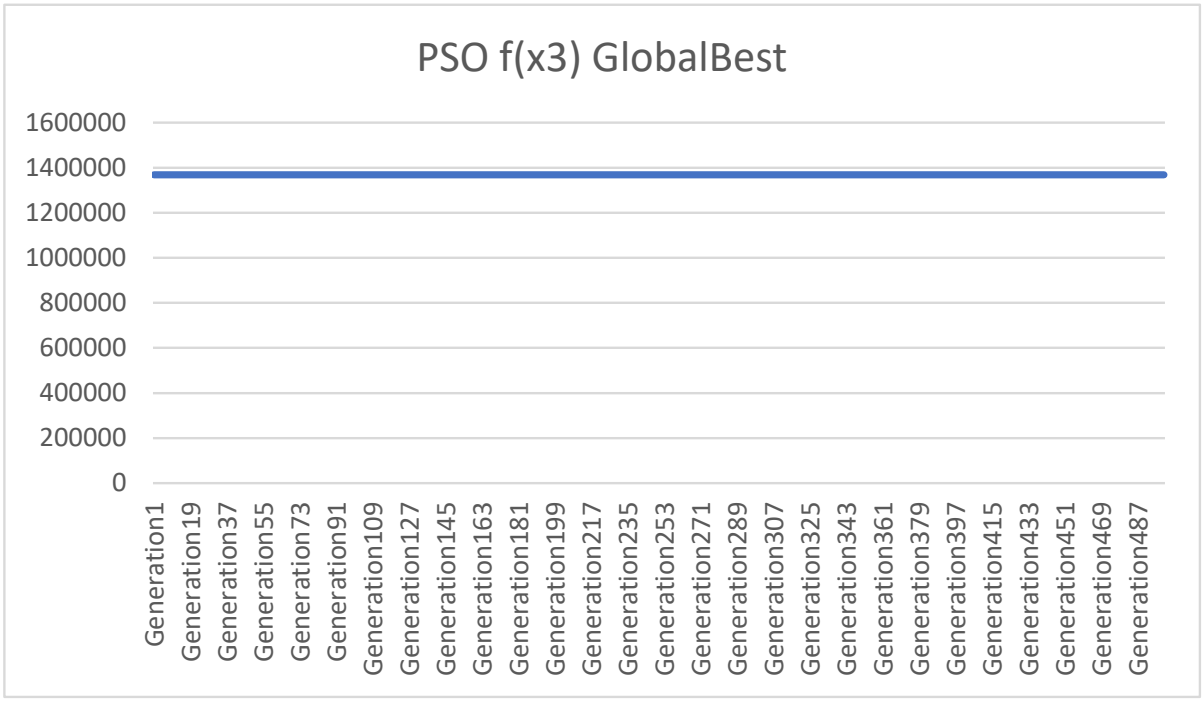


Figure 3.10: PSO Algorithm with Rastrigin Function

Function3.pdf

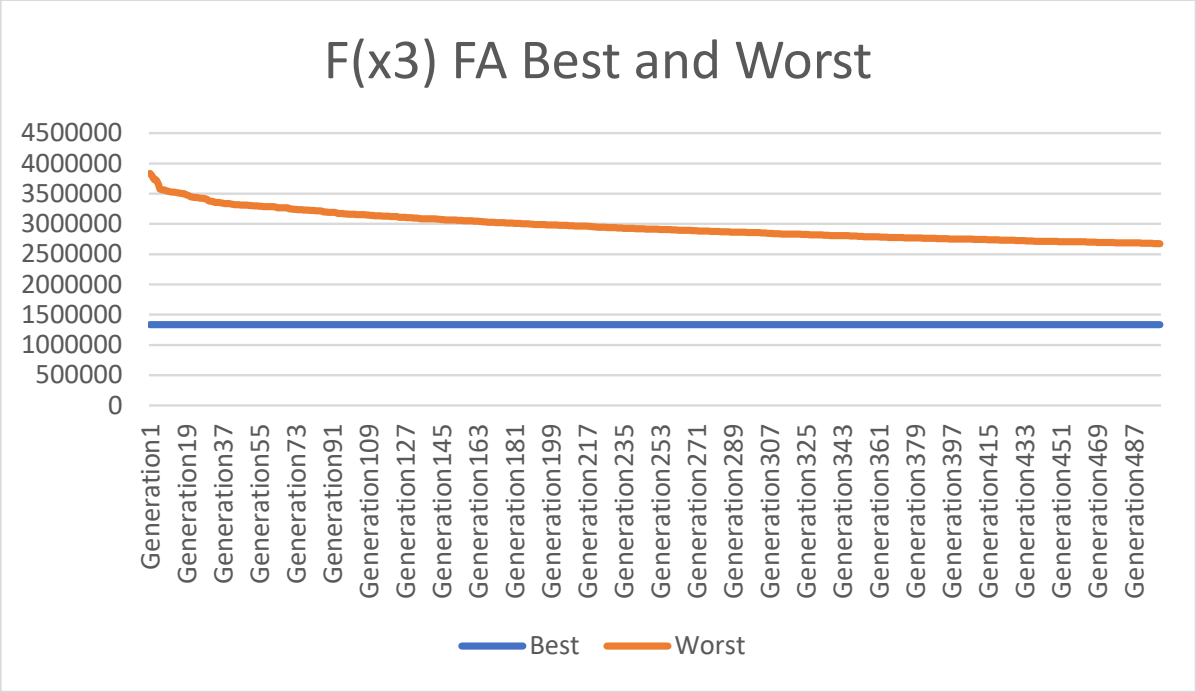


Figure 3.11: FA Algorithm with Rastrigin Function

Function3.pdf

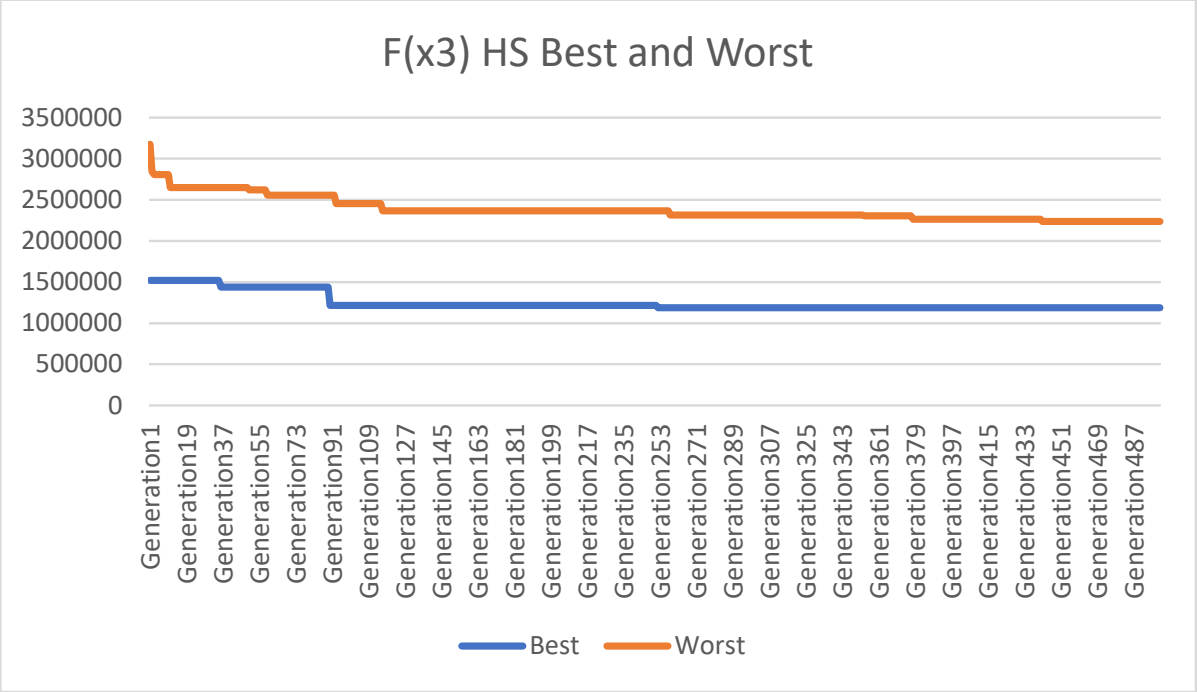


Figure 3.12: HS Algorithm with Rastrigin Function

Function4.pdf

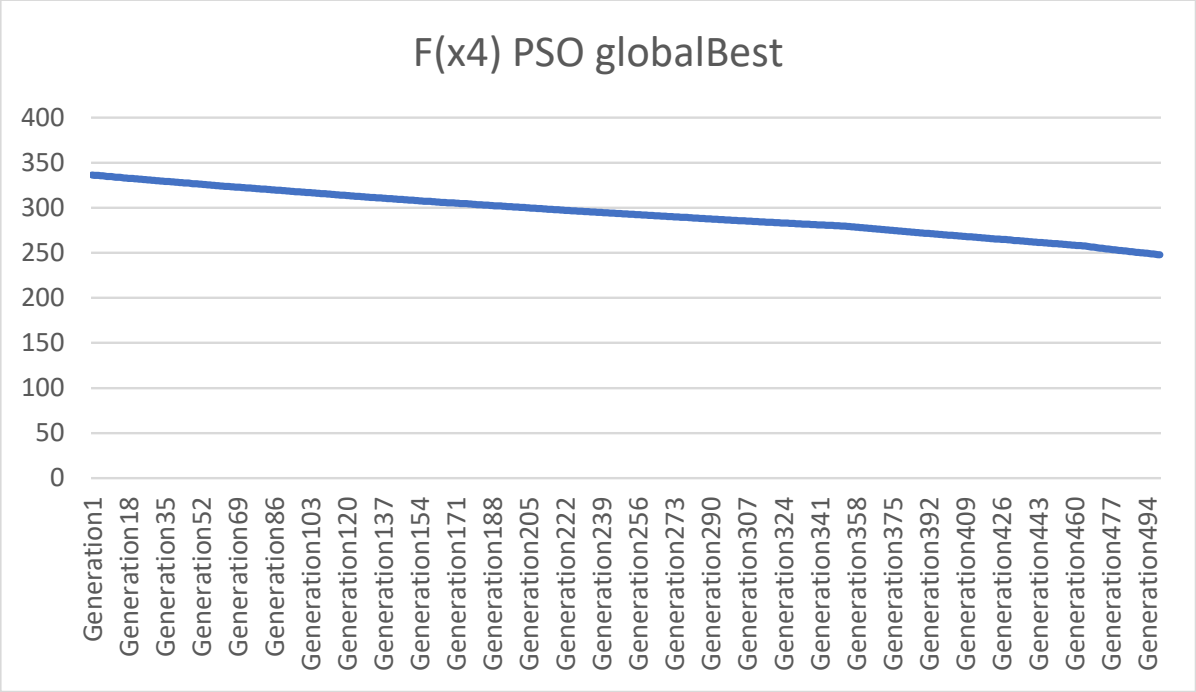


Figure 3.13: PSO Algorithm with Griewangk Function

Function4.pdf

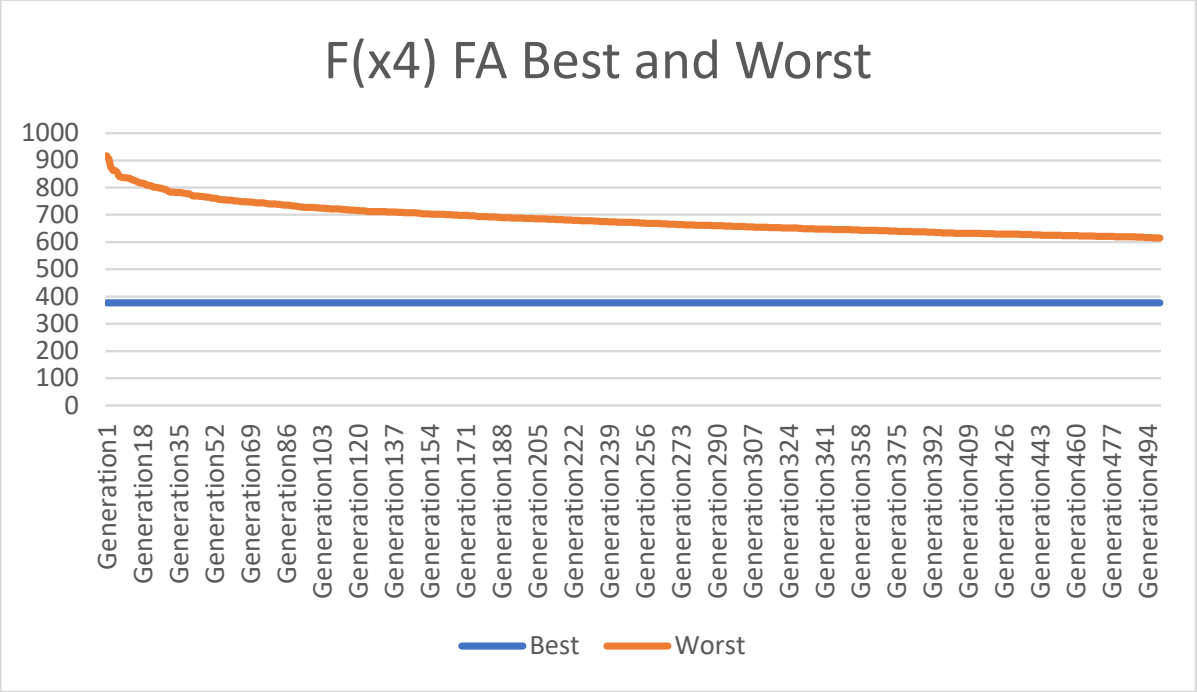


Figure 3.14: FA Algorithm with Griewangk Function

Function4.pdf

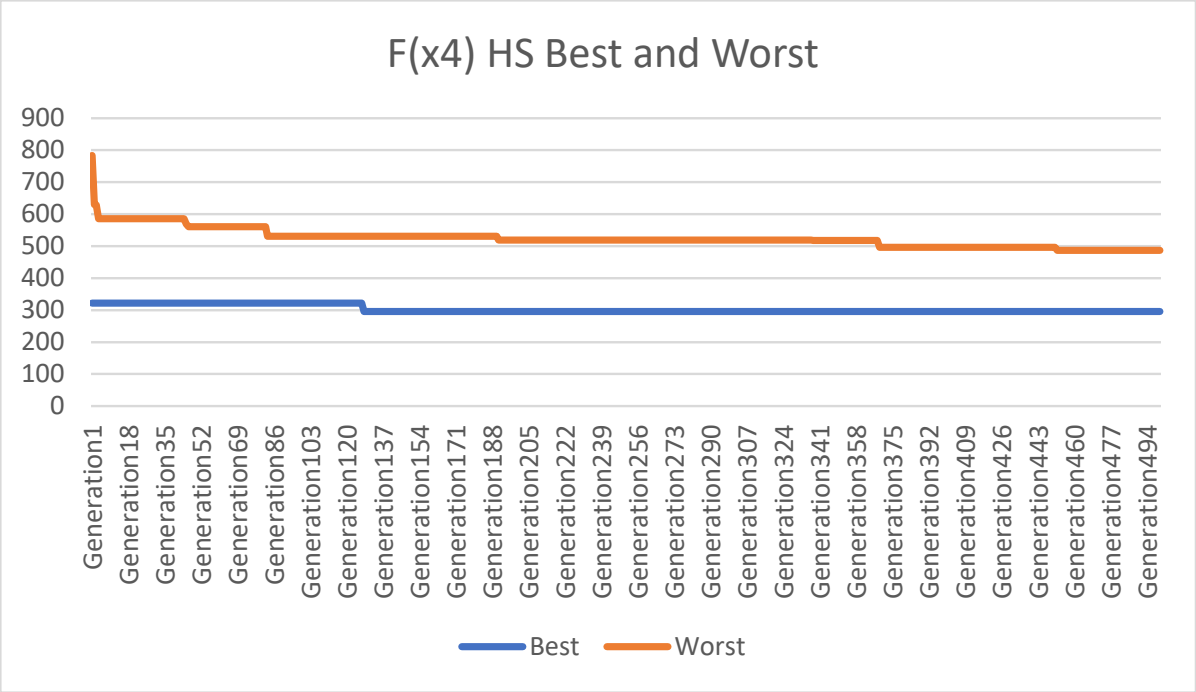


Figure 3.15: HS Algorithm with Griewangk Function

Function5.pdf

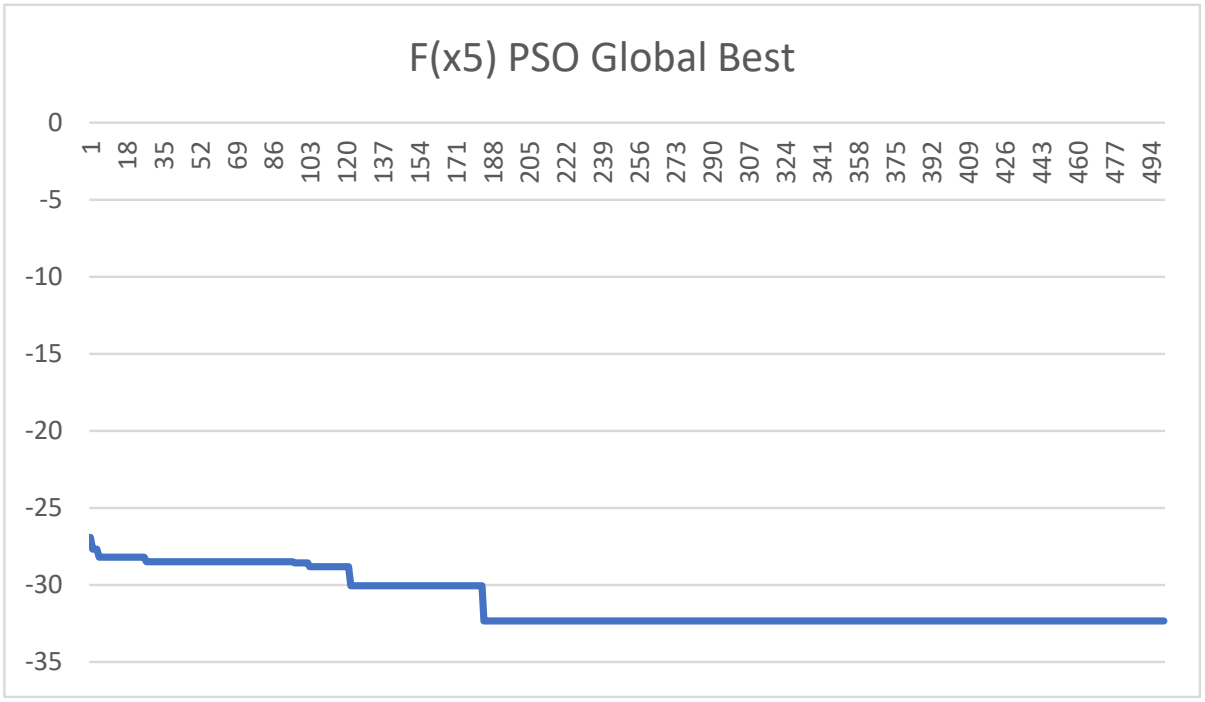


Figure 3.16: PSO Algorithm with SineEnvelope Function

Function5.pdf

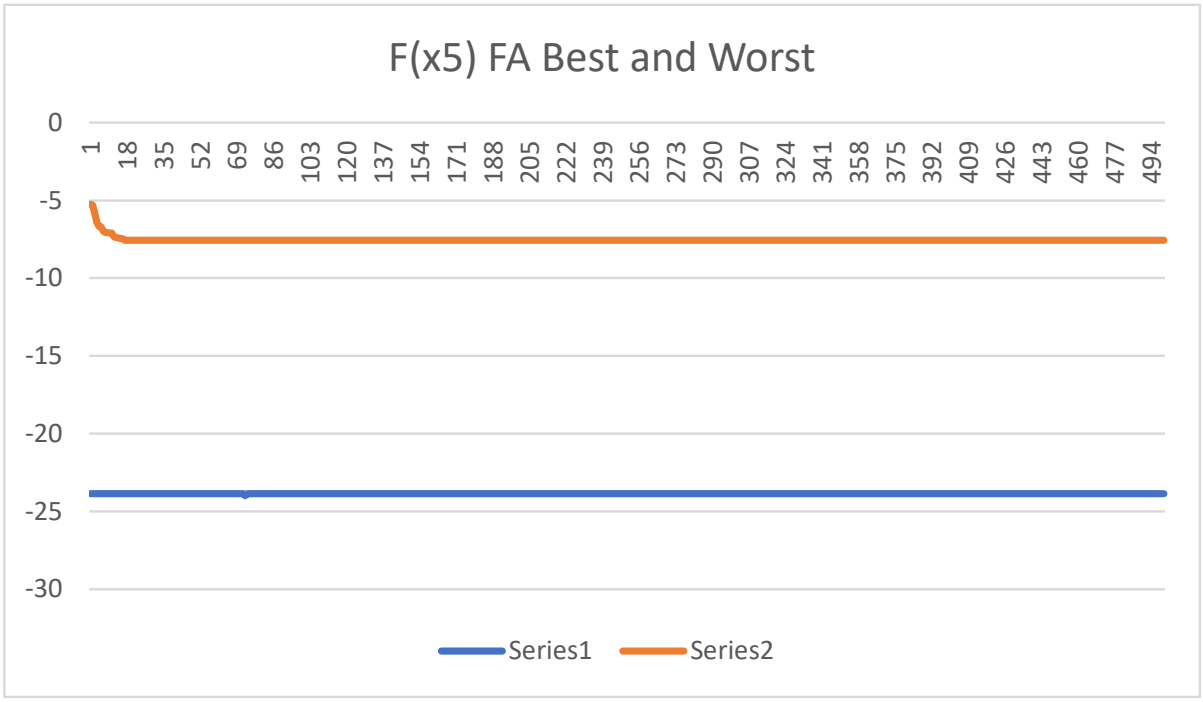


Figure 3.17: FA Algorithm with SineEnvelope Function

Function5.pdf

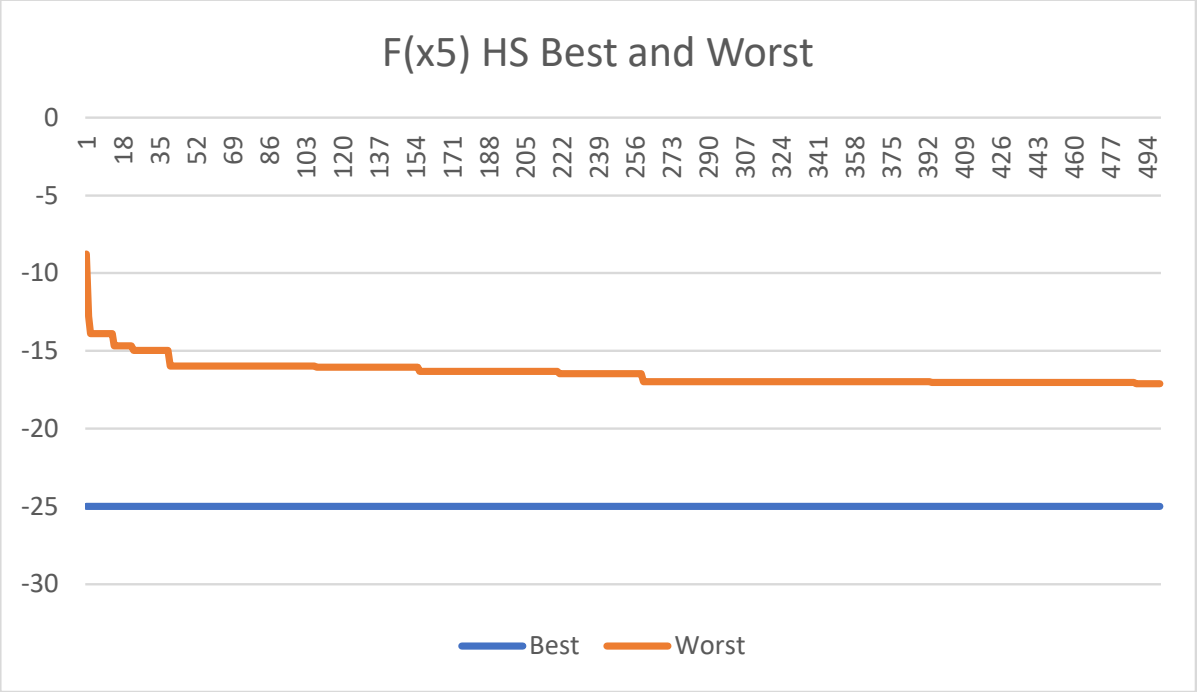


Figure 3.18: HS Algorithm with SineEnvelope Function

Function5.pdf

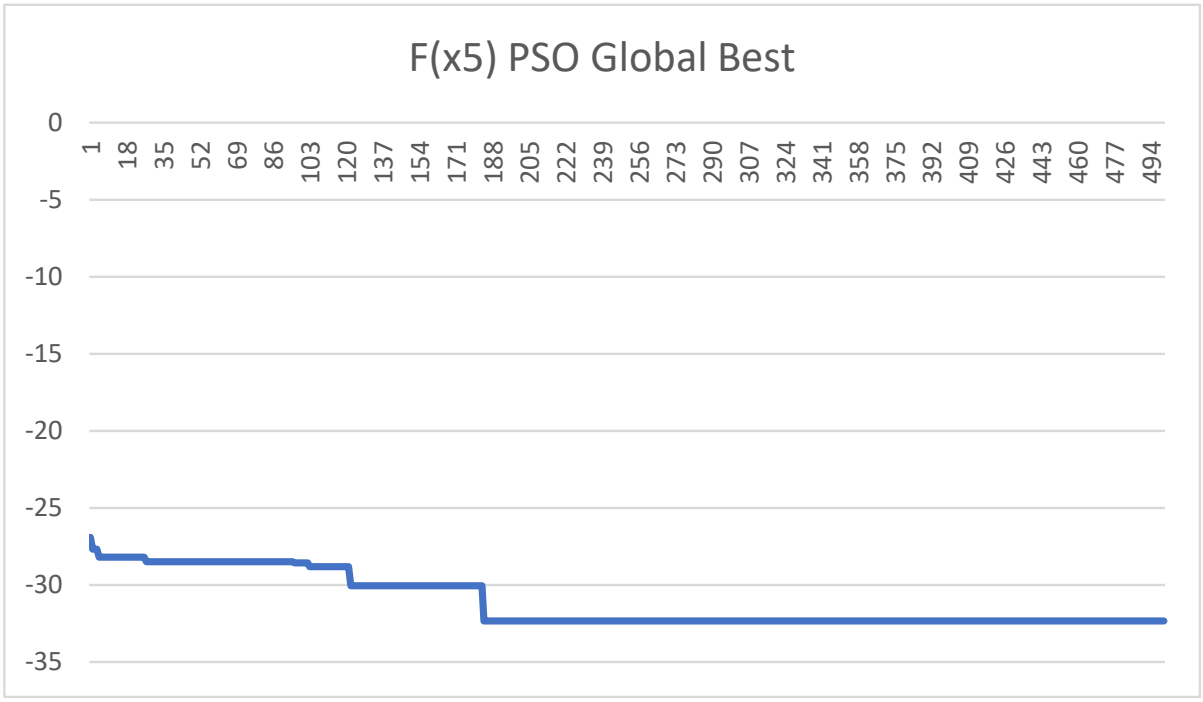


Figure 3.19: PSO Algorithm with StretchedV Function

Function6.pdf

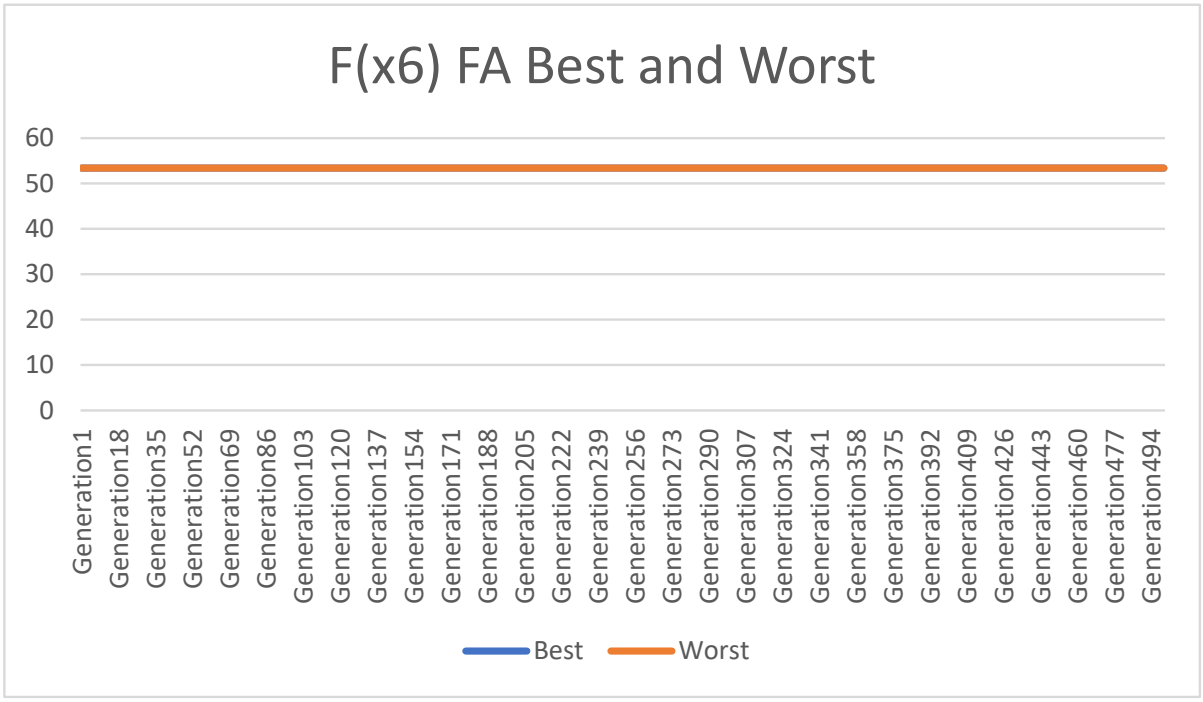


Figure 3.20: FA Algorithm with StretchedV Function

Function6.pdf

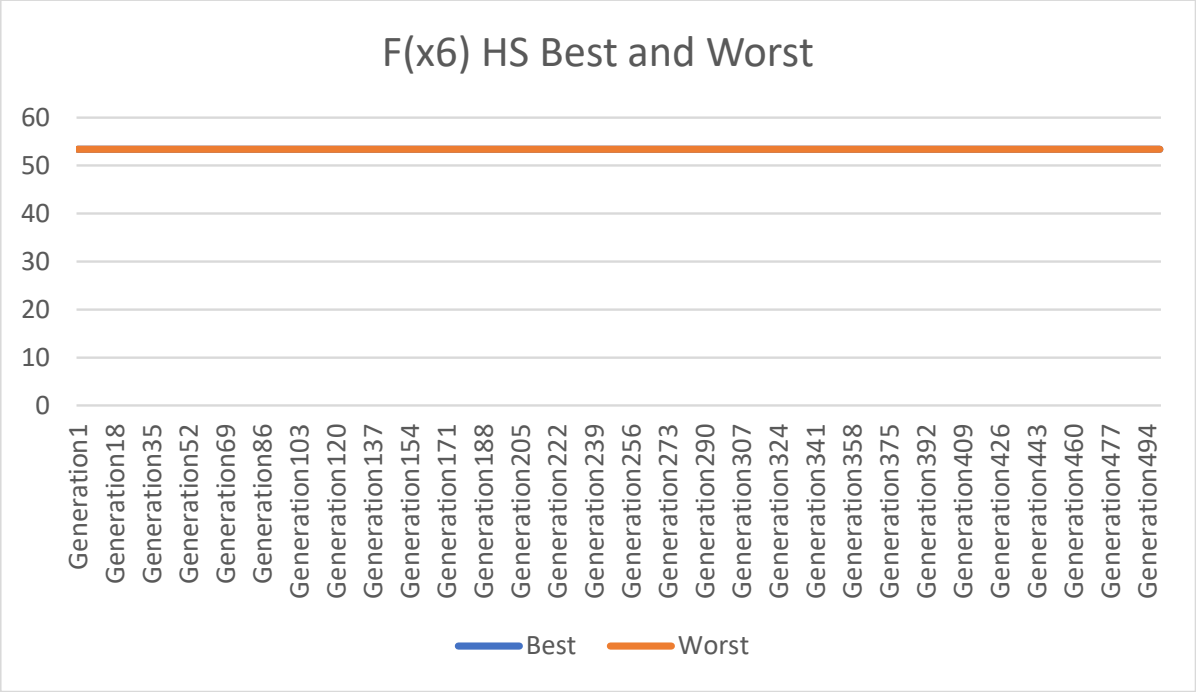


Figure 3.21: HS Algorithm with StretchedV Function

Function7.pdf

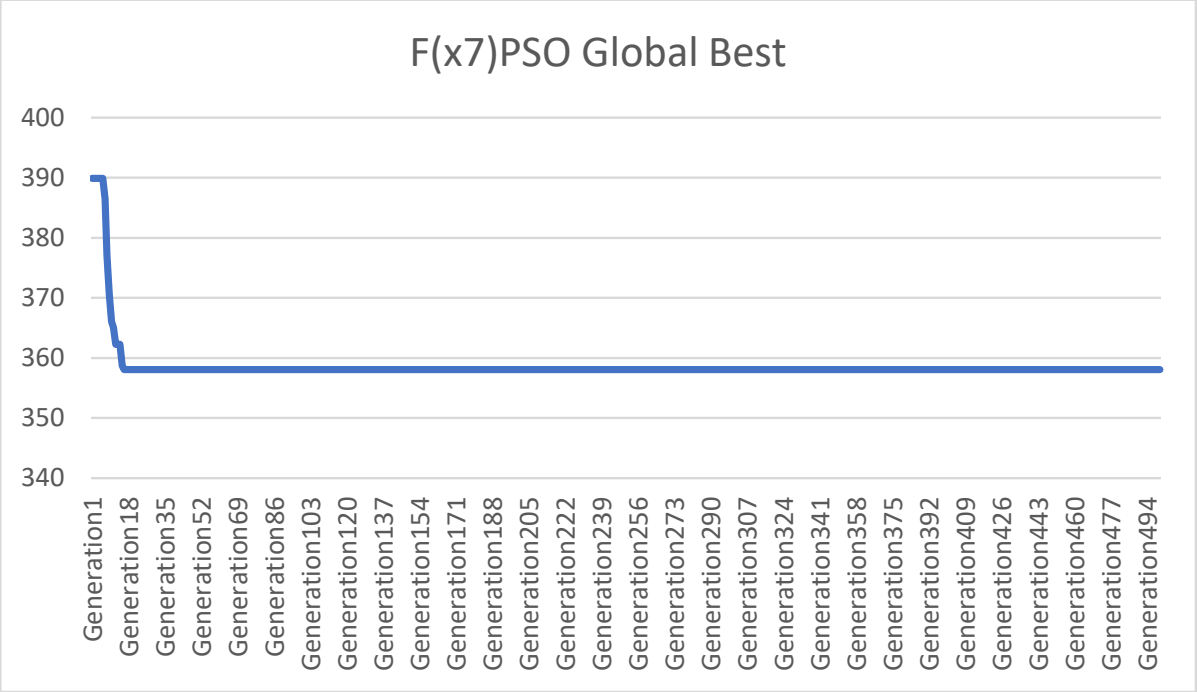


Figure 3.22: PSO Algorithm with Ackley1 Function

Function7.pdf

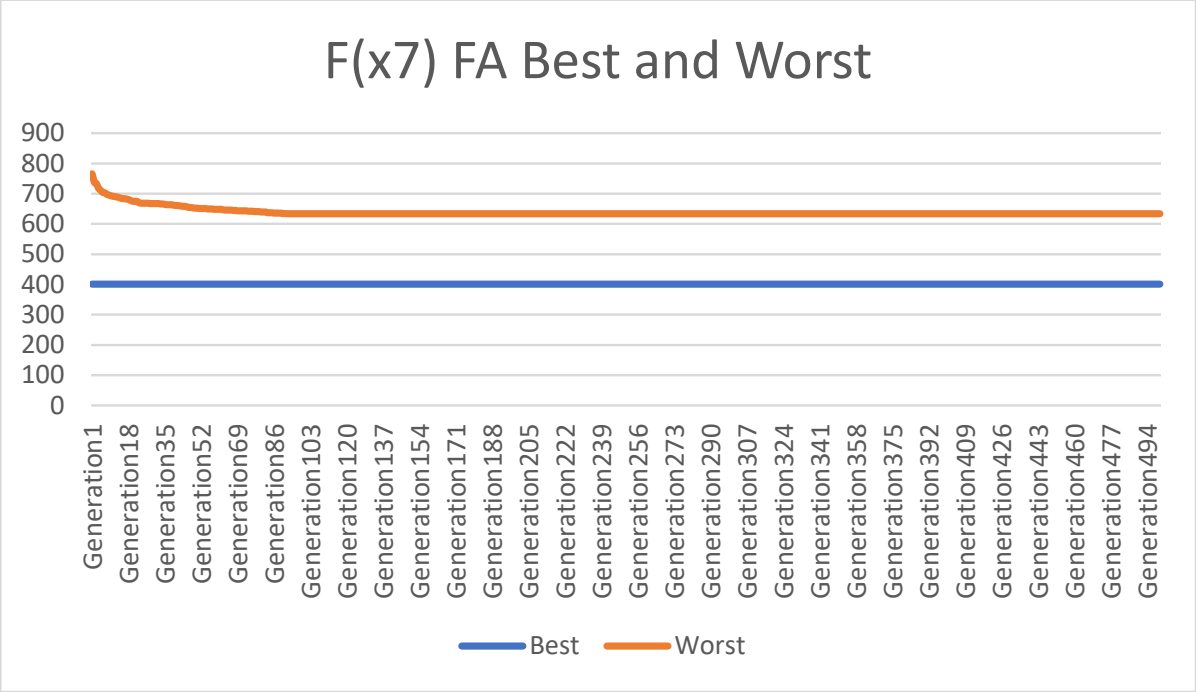


Figure 3.23: FA Algorithm with Ackley1 Function

Function7.pdf

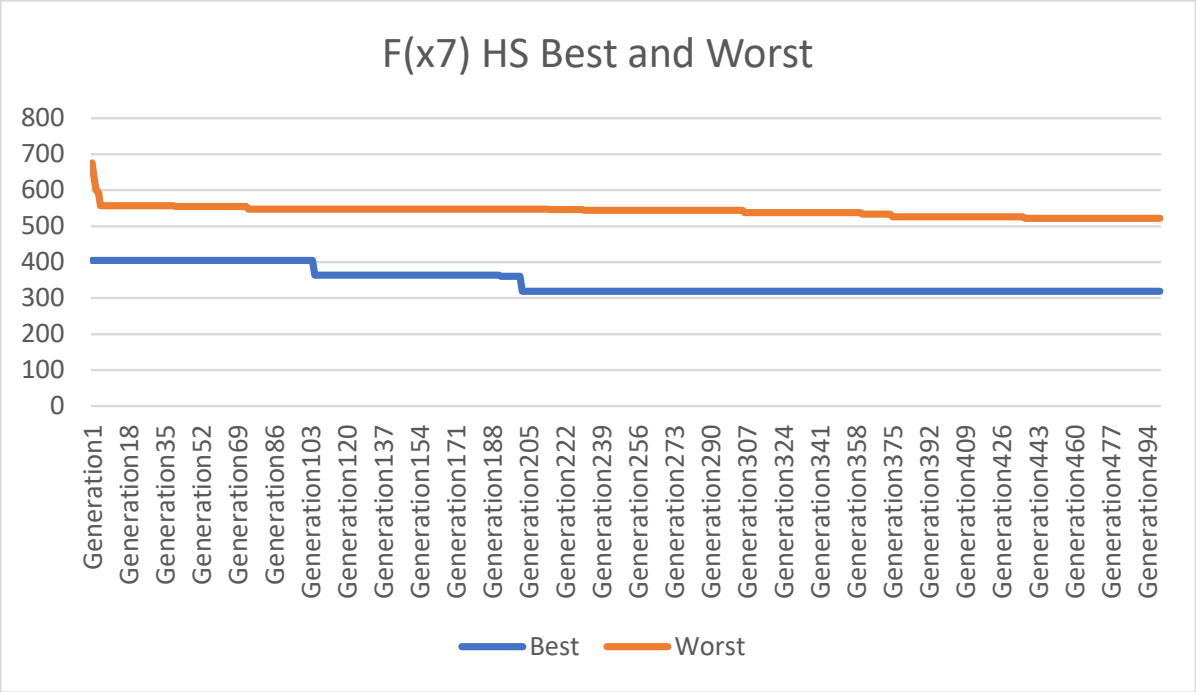


Figure 3.24: HS Algorithm with Ackley1 Function

Function8.pdf

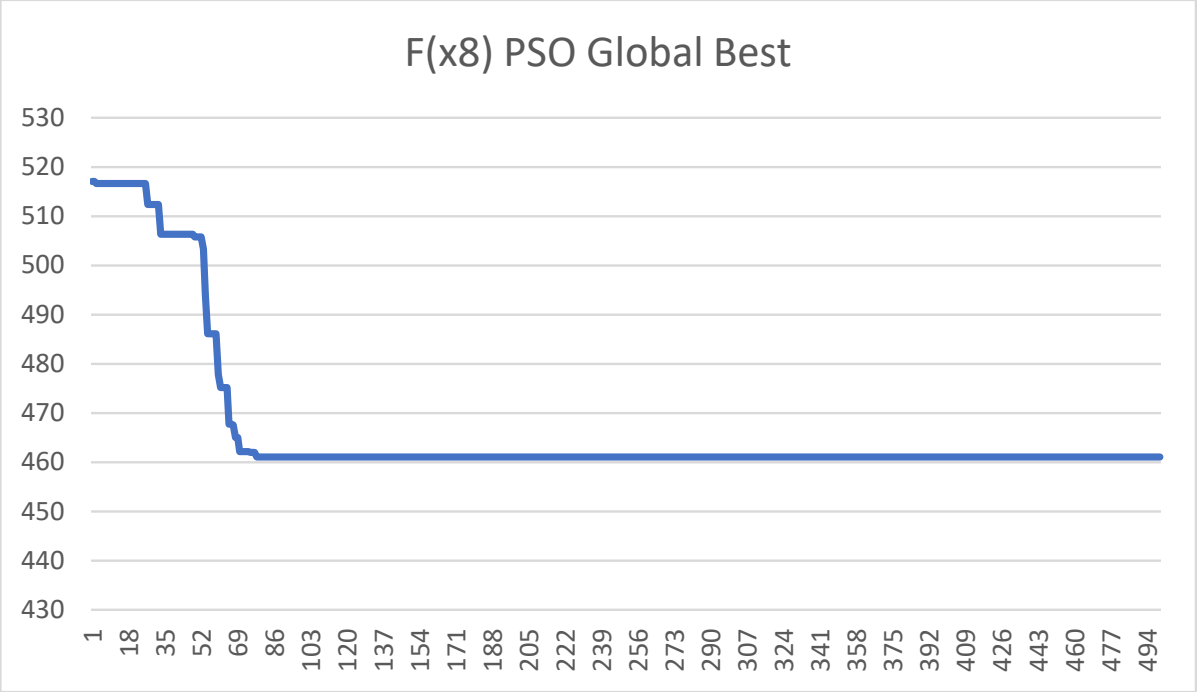


Figure 3.25: PSO Algorithm with Ackely2 Function

Function8.pdf

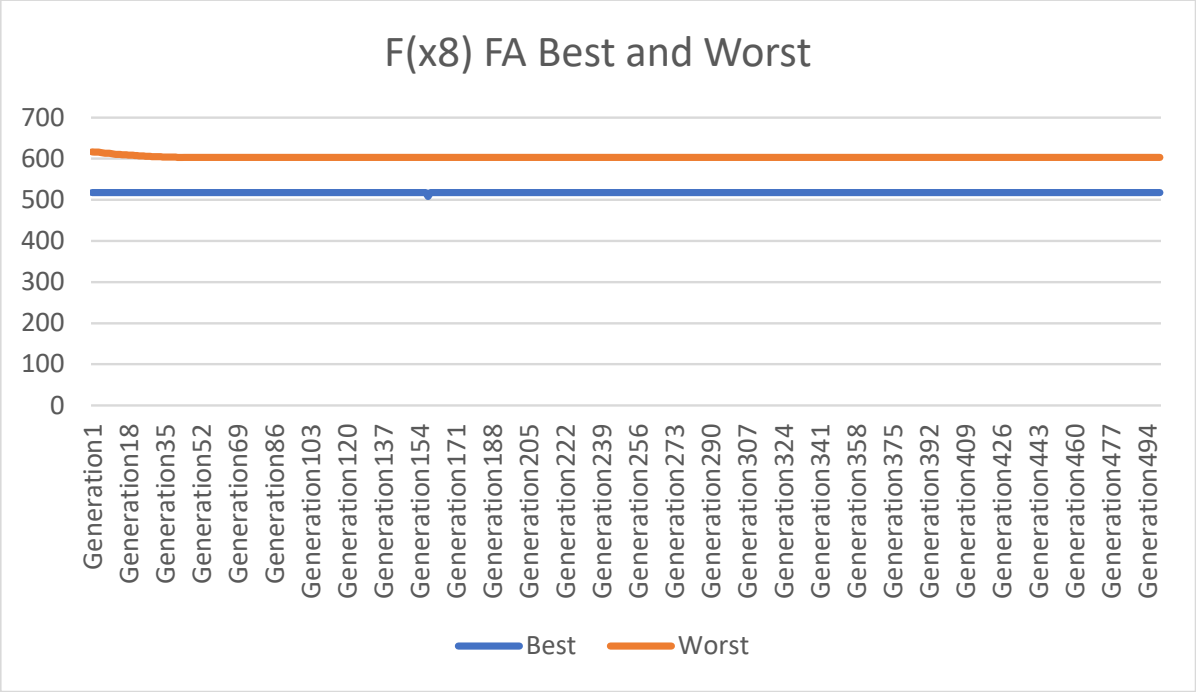


Figure 3.26: FA Algorithm with Ackely2 Function

Function8.pdf

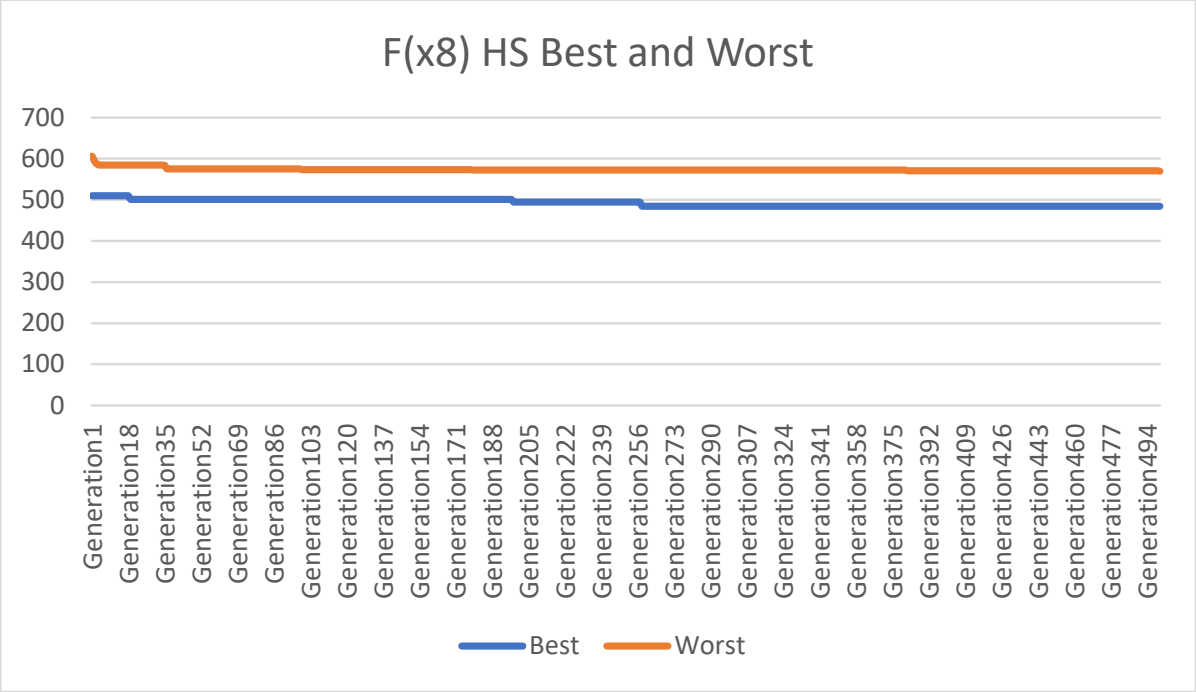


Figure 3.27: HS Algorithm with Ackely2 Function

Function9.pdf

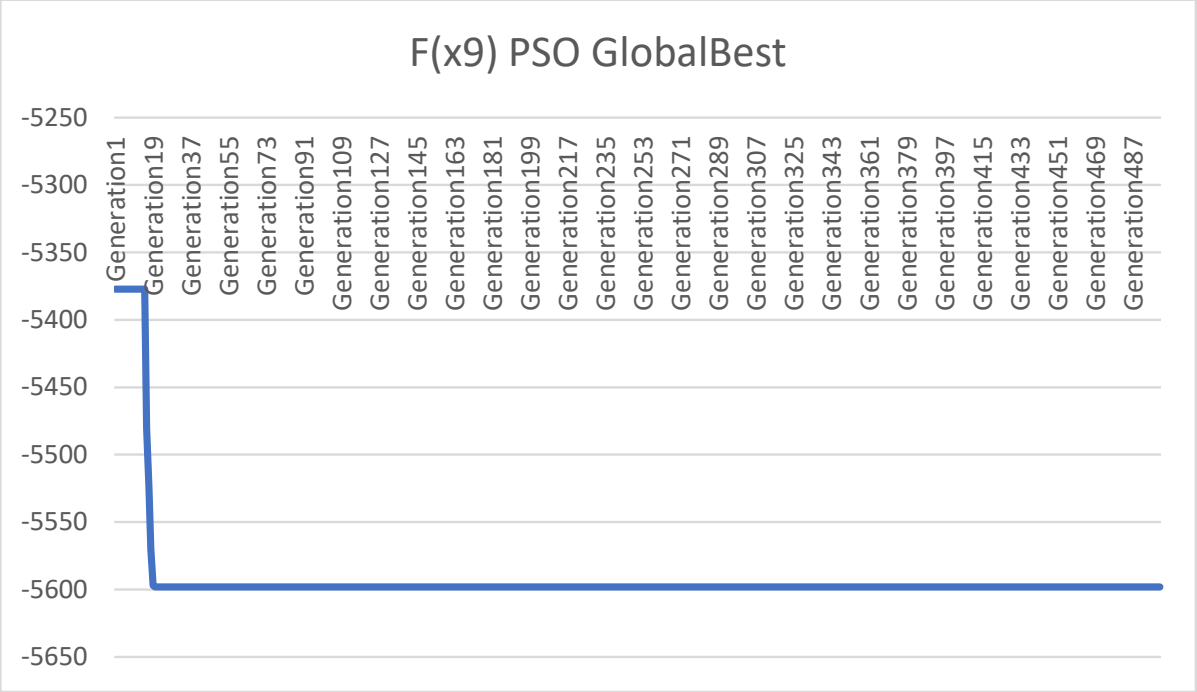


Figure 3.28: PSO Algorithm with EggHolder Function

Function9.pdf

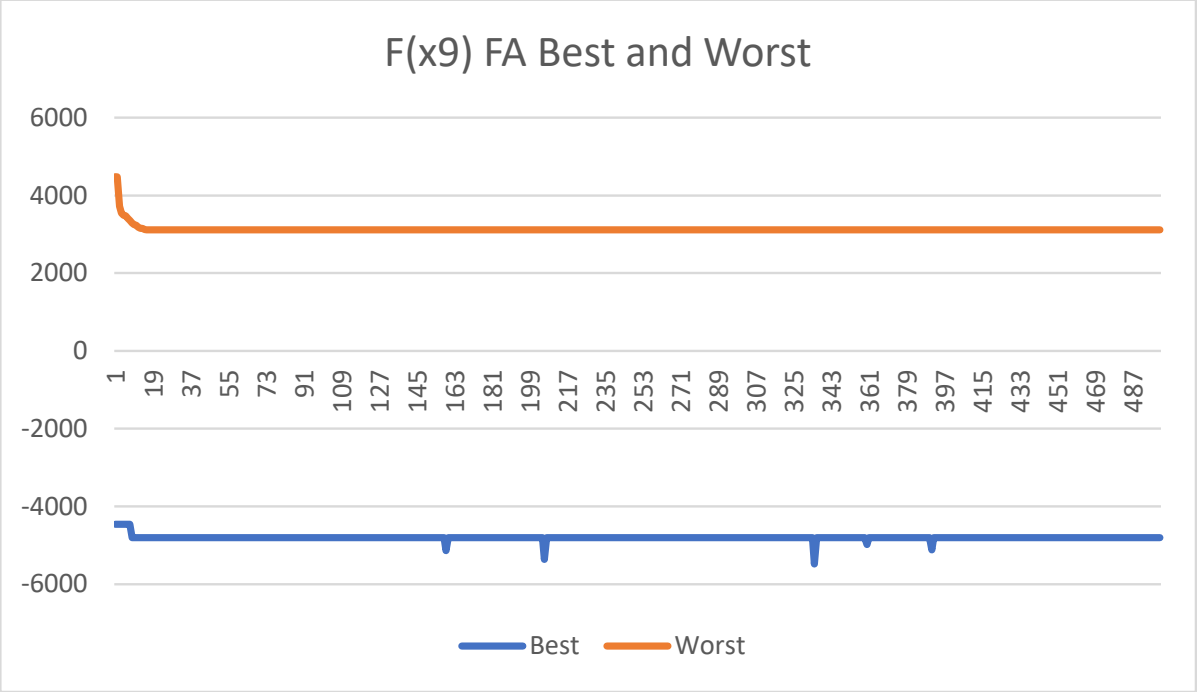


Figure 3.29: FA Algorithm with EggHolder Function

Function9.pdf

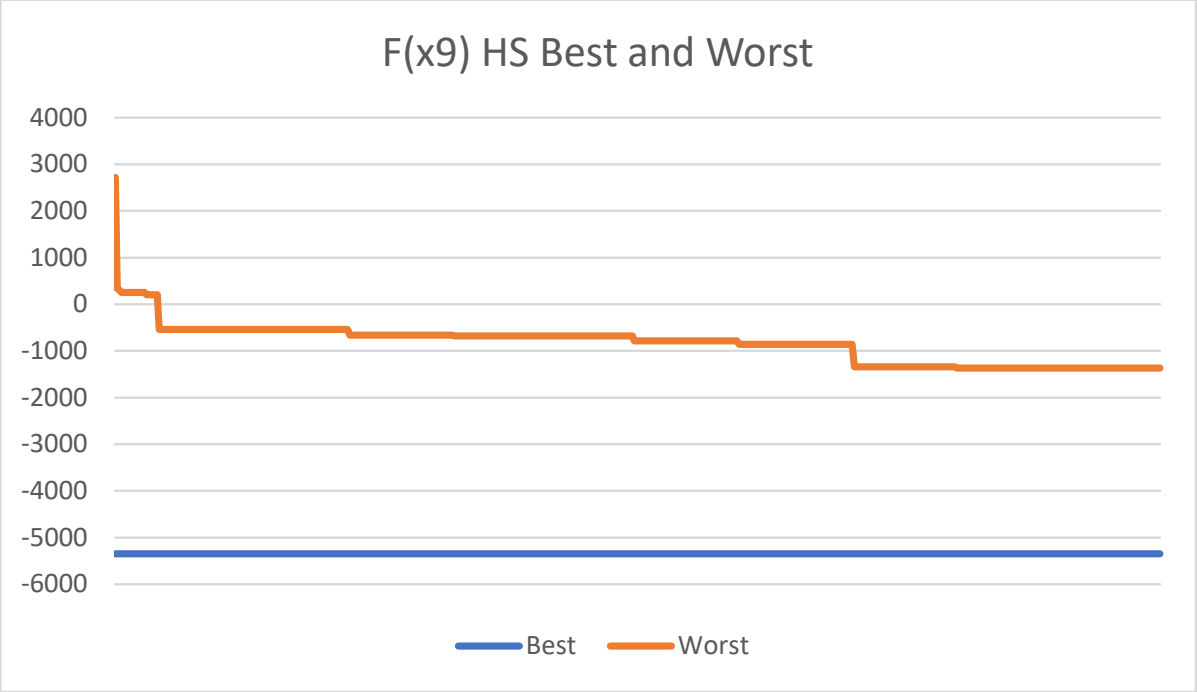


Figure 3.30: HS Algorithm with EggHolder Function

Function10.pdf

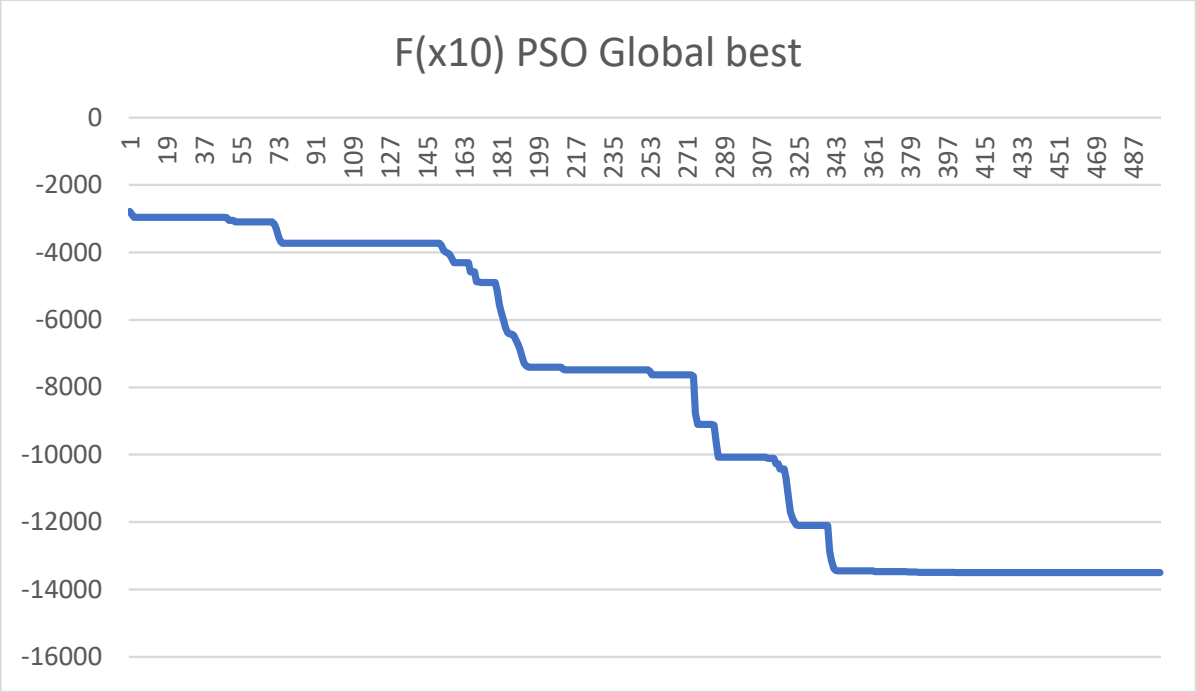


Figure 3.31: PSO Algorithm with Rana Function

Function10.pdf

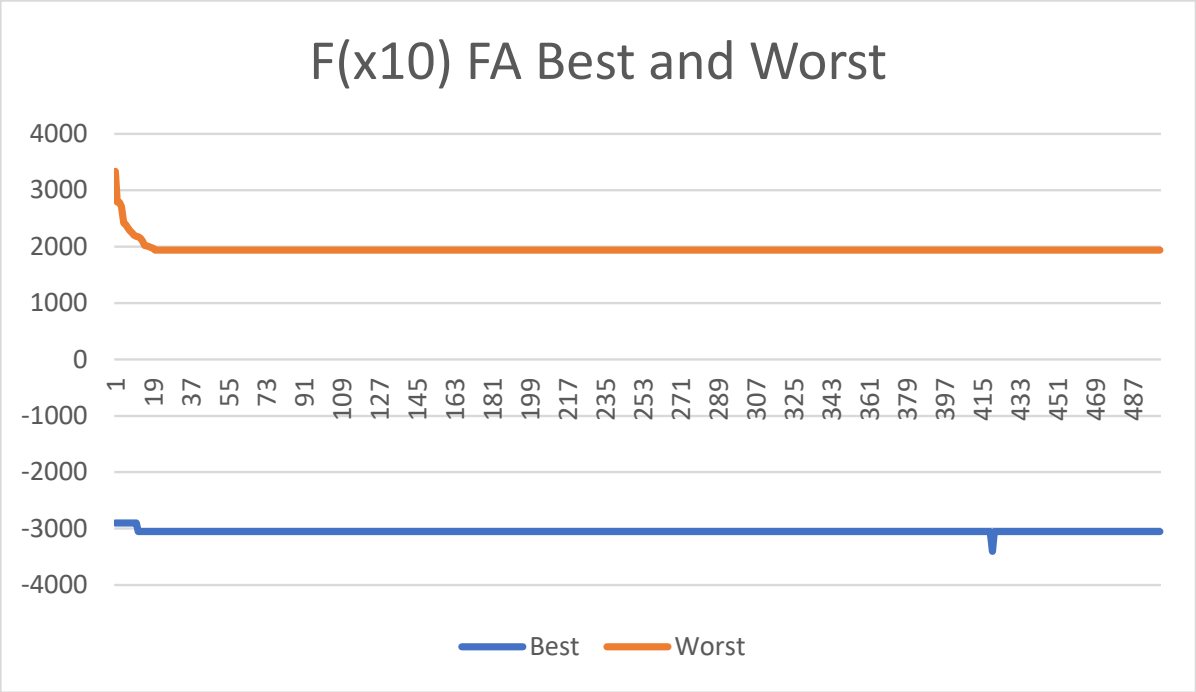


Figure 3.32: FA Algorithm with Rana Function

Function10.pdf

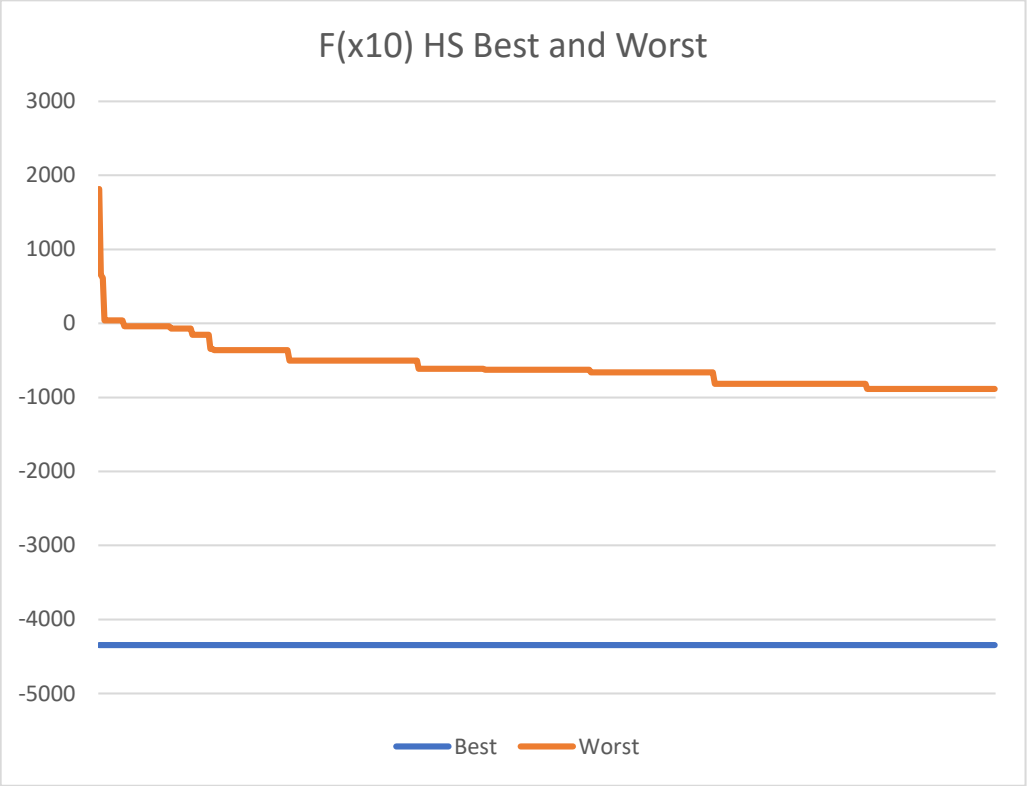


Figure 3.33: HS Algorithm with Rana Function

Function11.pdf

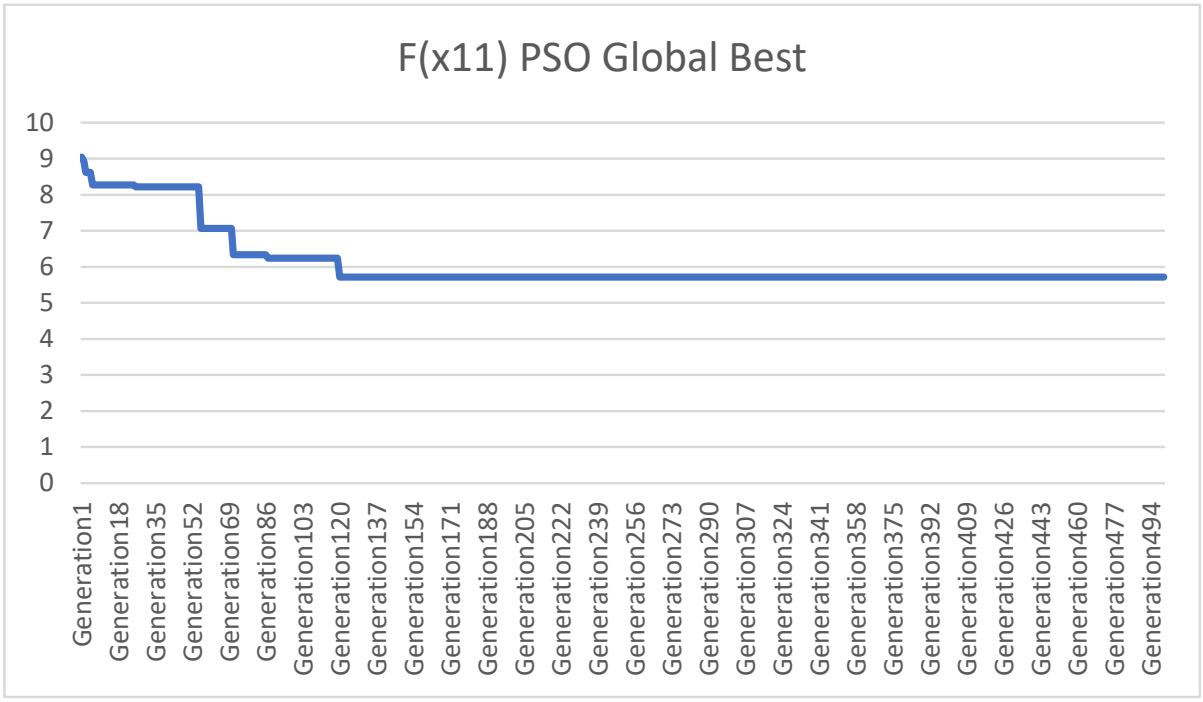


Figure 3.34: PSO Algorithm with Pathological Function

Function11.pdf

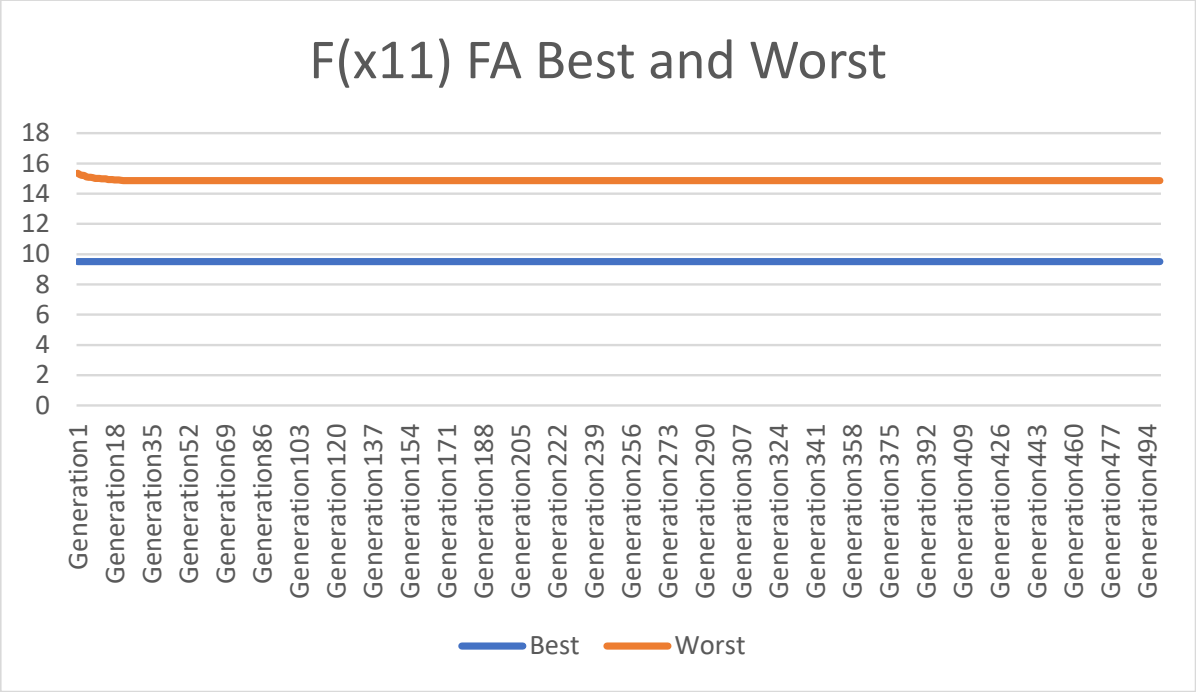


Figure 3.35: FA Algorithm with Pathological Function

Function11.pdf

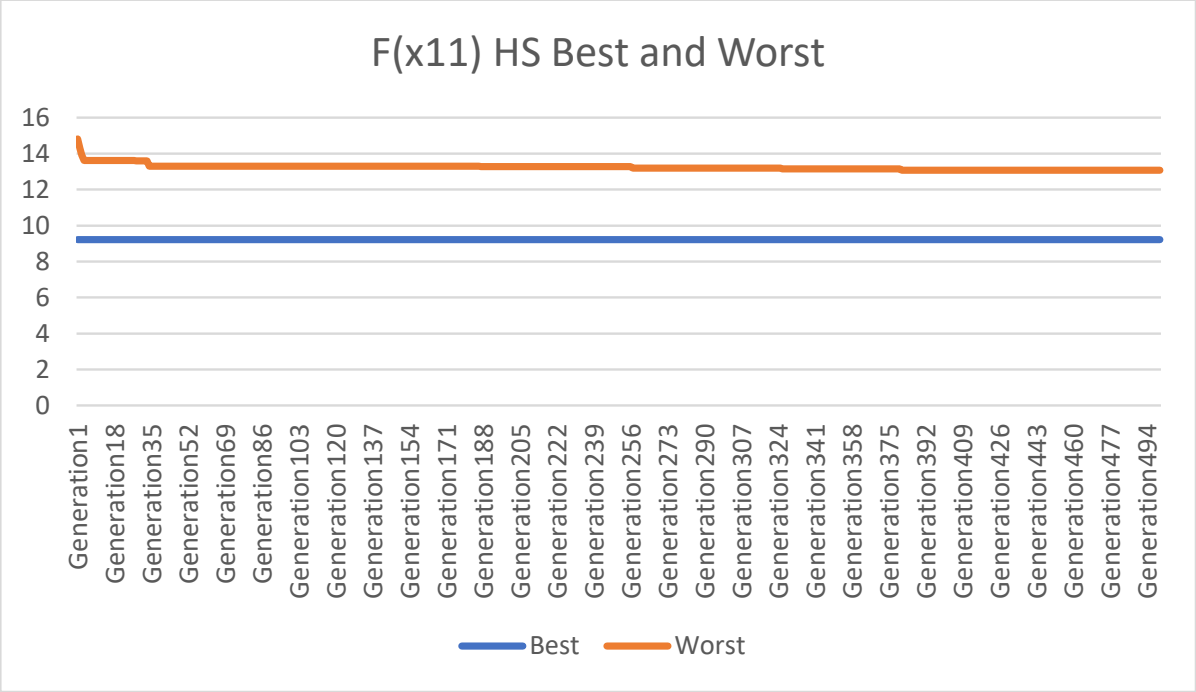


Figure 3.36: HS Algorithm with Pathological Function

Function12.pdf

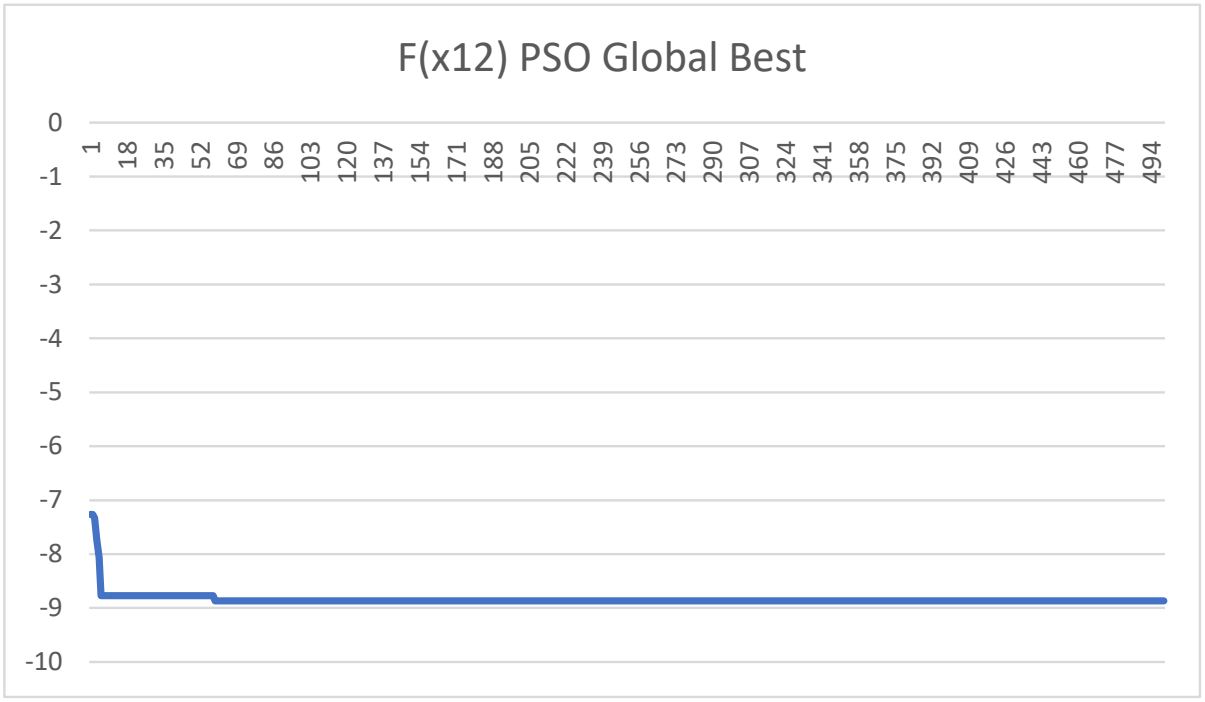


Figure 3.37: PSO Algorithm with Michalewicz Function

Function12.pdf

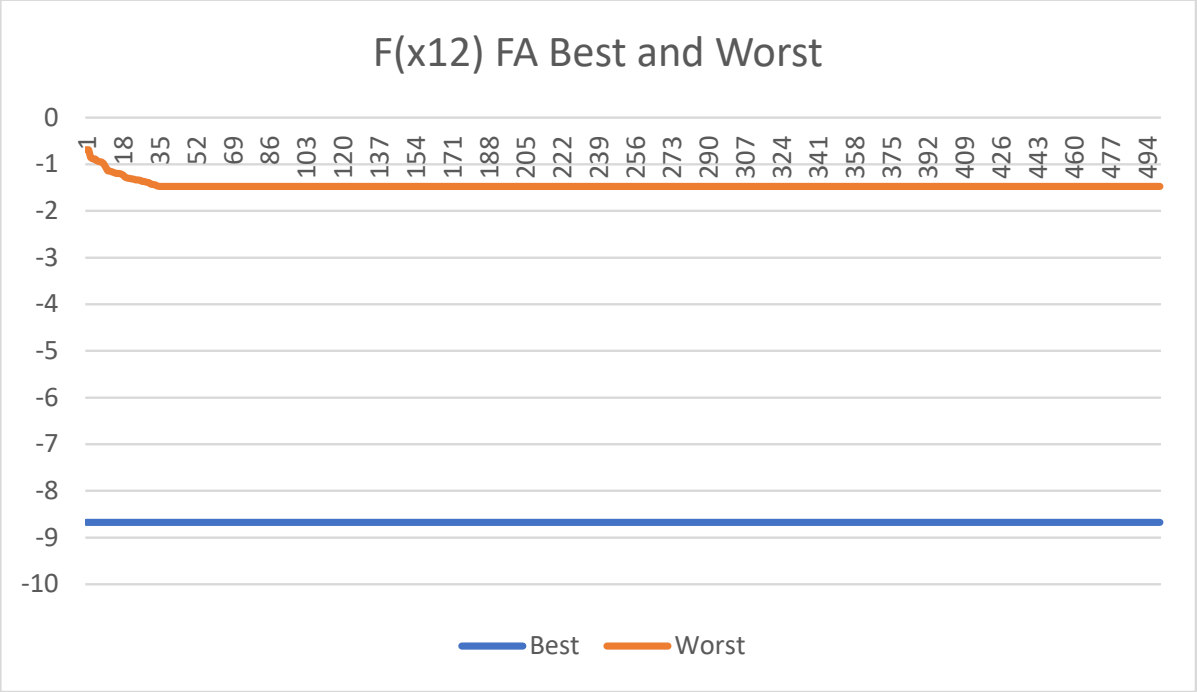


Figure 3.38: FA Algorithm with Michalewicz Function

Function12.pdf

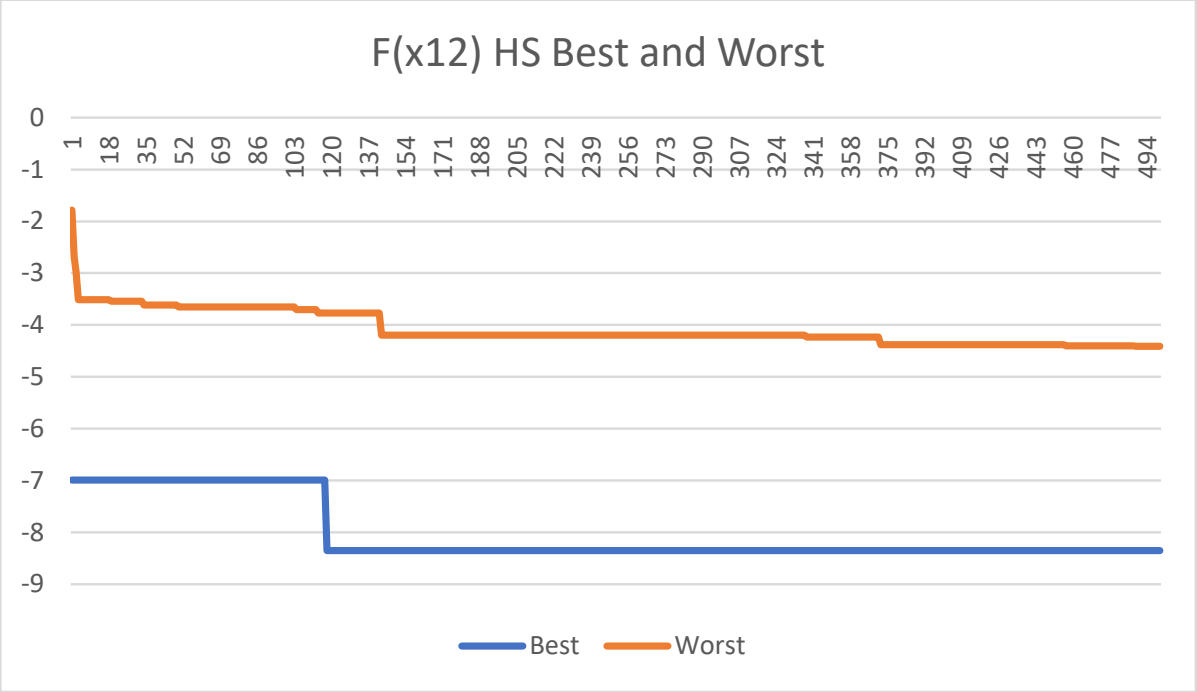


Figure 3.39: HS Algorithm with Michalewicz Function

Function13.pdf

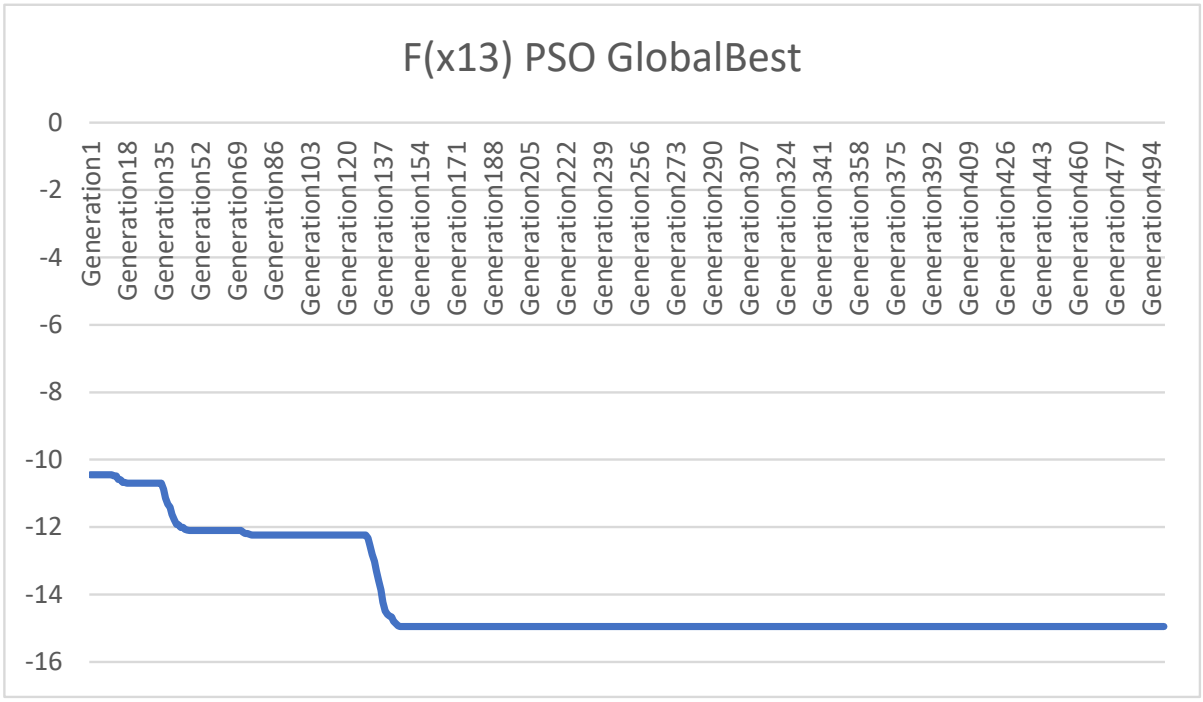


Figure 3.40: PSO Algorithm with Masters Function

Function13.pdf

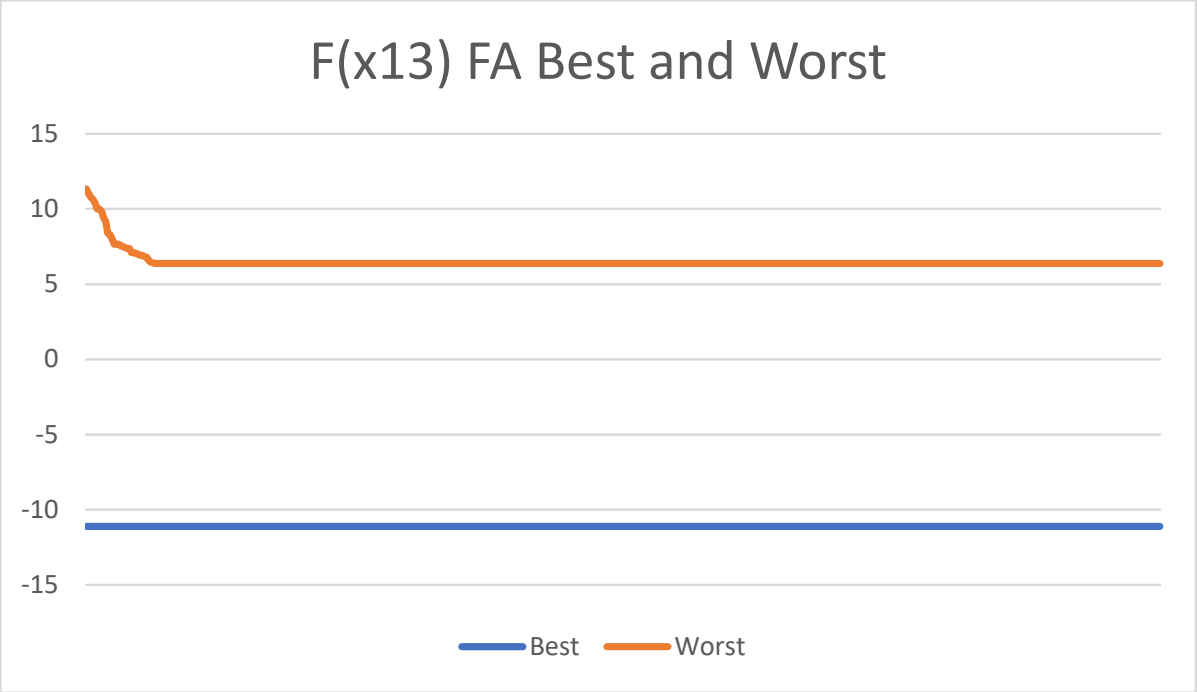


Figure 3.41: FA Algorithm with Masters Function

Function13.pdf

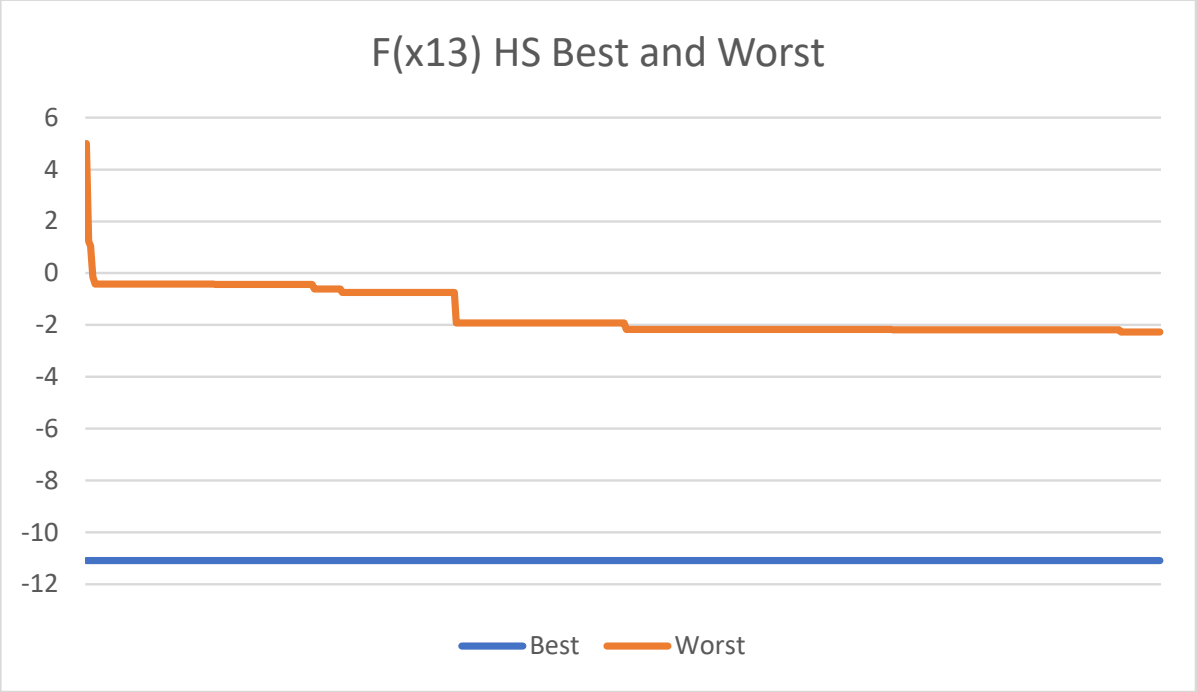


Figure 3.42: HS Algorithm with Masters Function

Function14.pdf

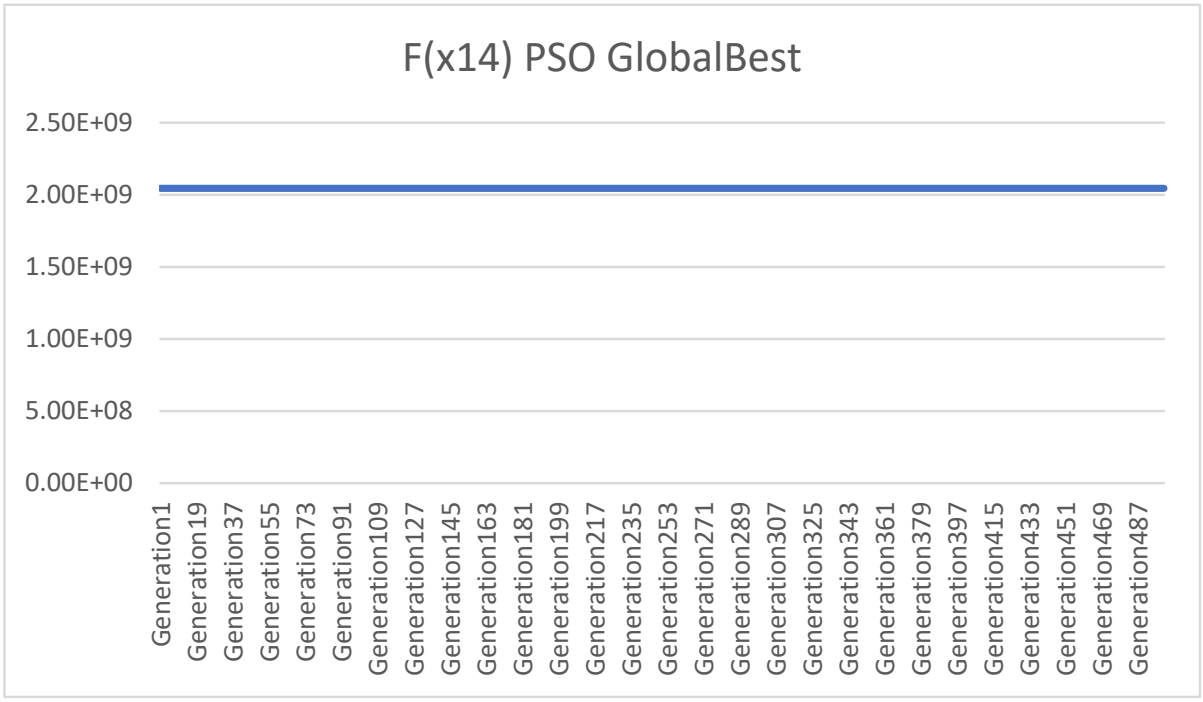


Figure 3.43: PSO Algorithm with Quartic Function

Function14.pdf

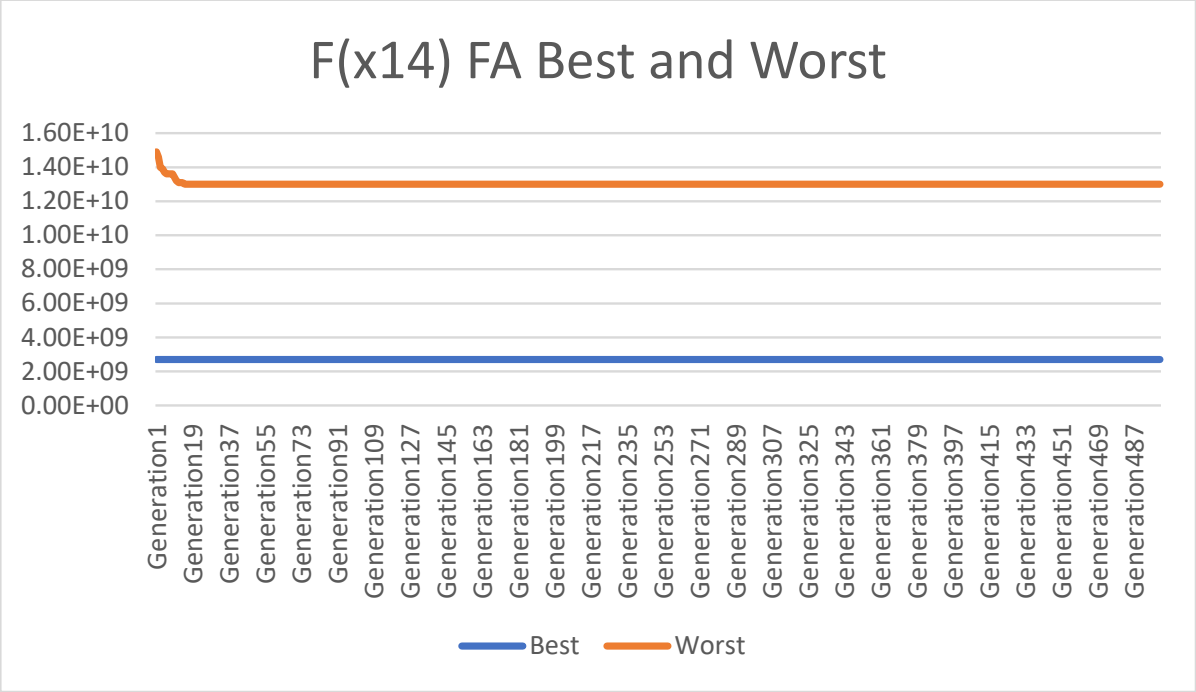


Figure 3.44: FA Algorithm with Quartic Function

Function14.pdf

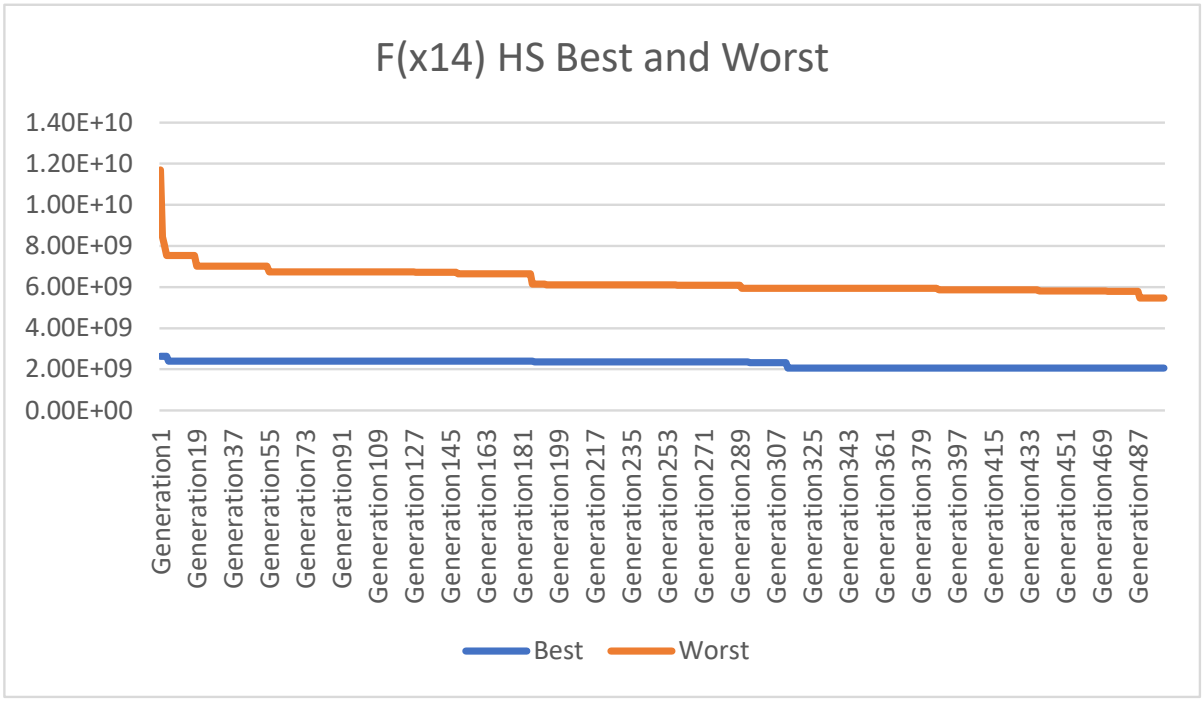


Figure 3.45: HS Algorithm with Quartic Function

Function15.pdf

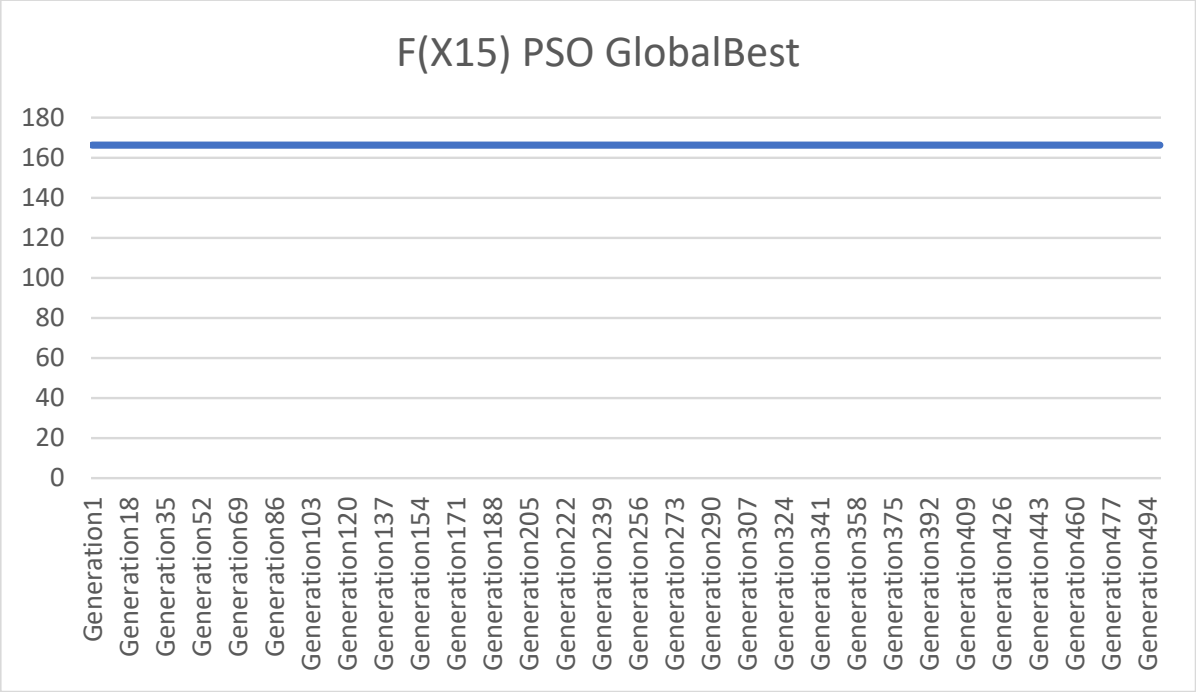


Figure 3.46: PSO Algorithm with Levy Function

Function15.pdf

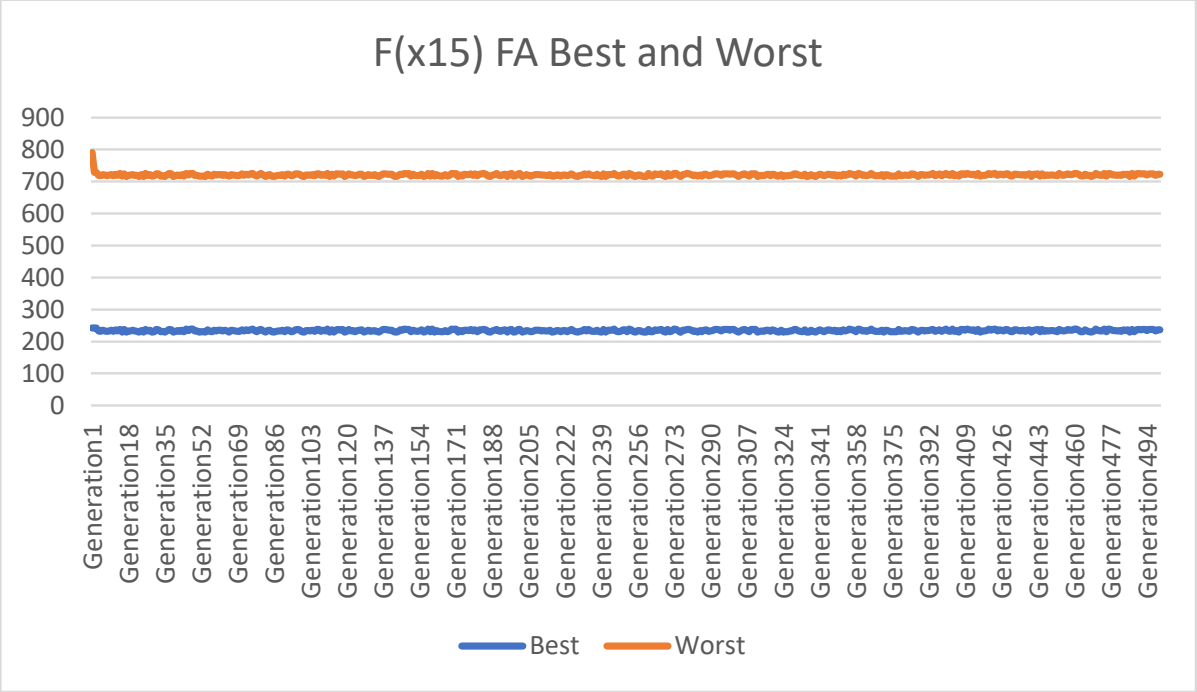


Figure 3.47: FA Algorithm with Levy Function

Function15.pdf

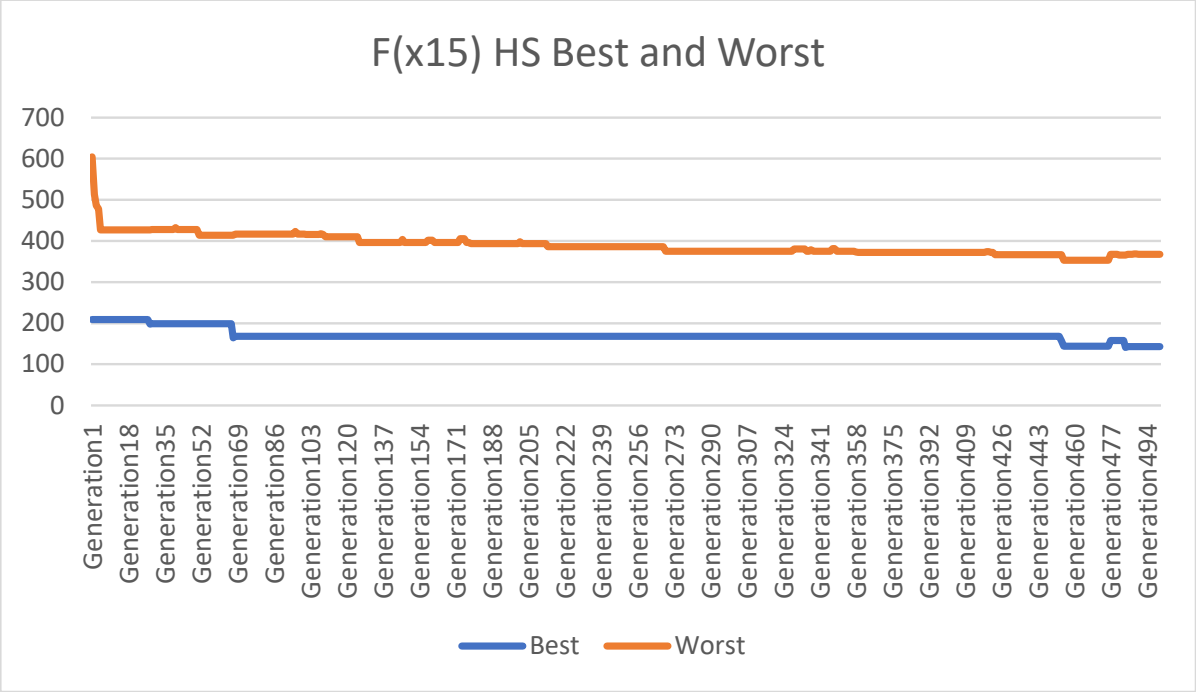


Figure 3.48: HS Algorithm with Levy Function

Function16.pdf

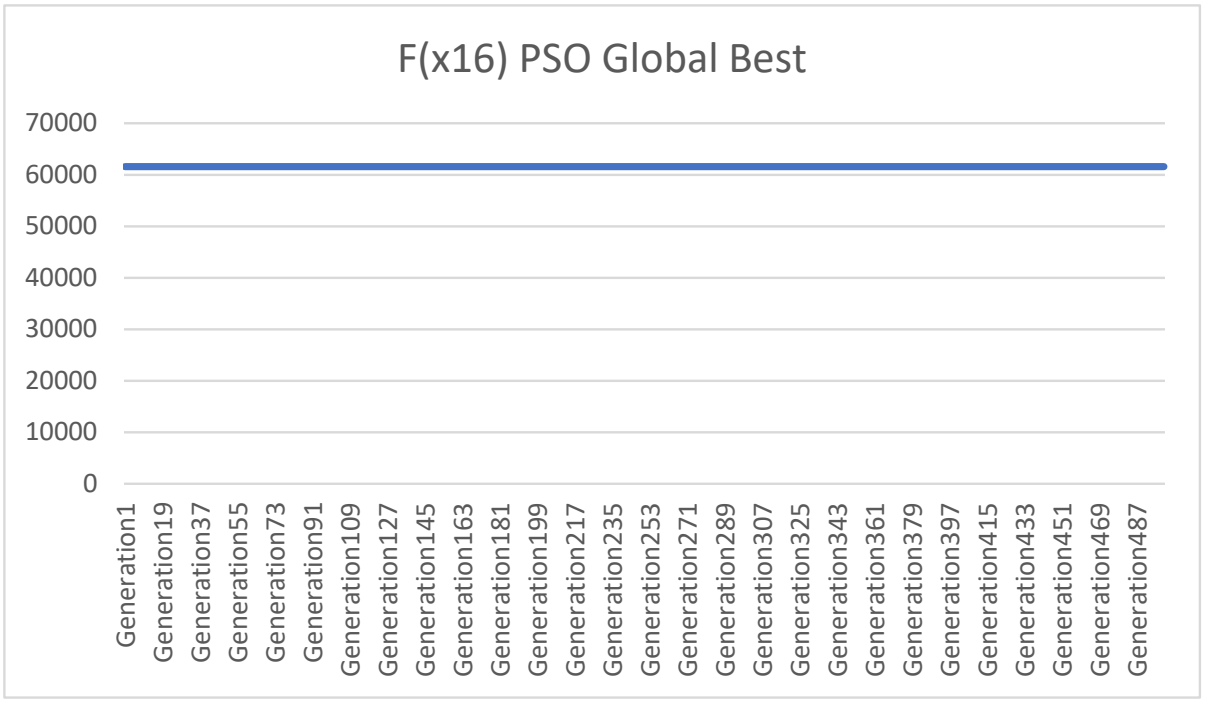


Figure 3.49: PSO Algorithm with Step Function

Function16.pdf

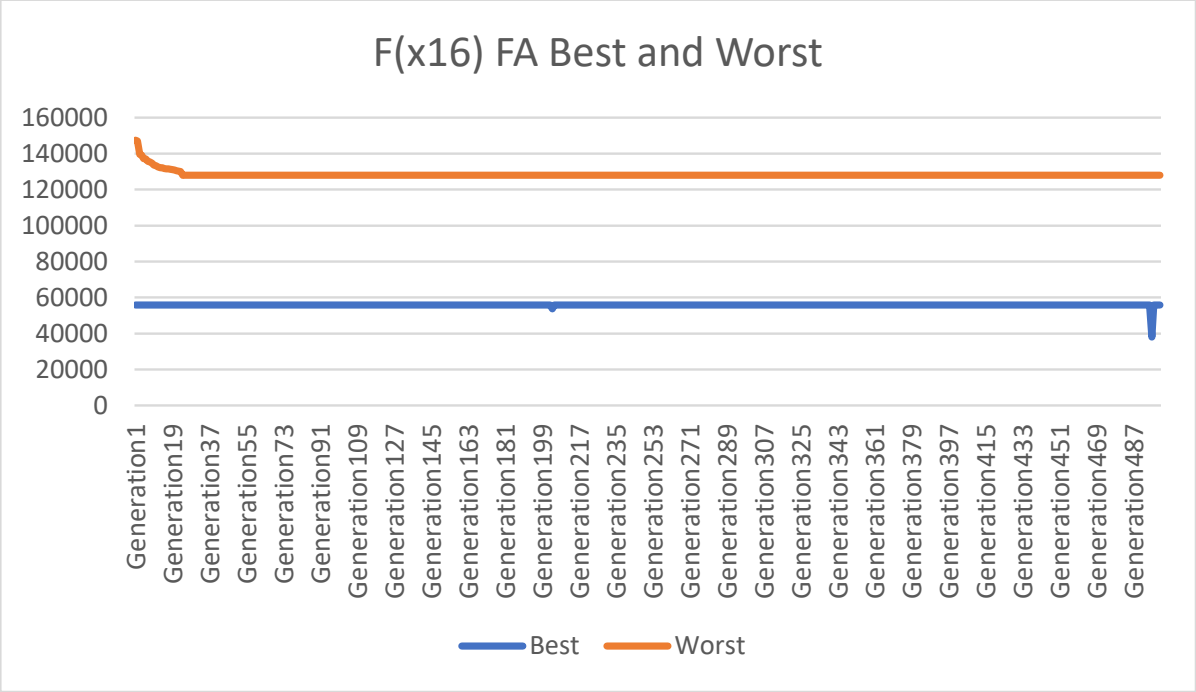


Figure 3.50: FA Algorithm with Step Function

Function16.pdf

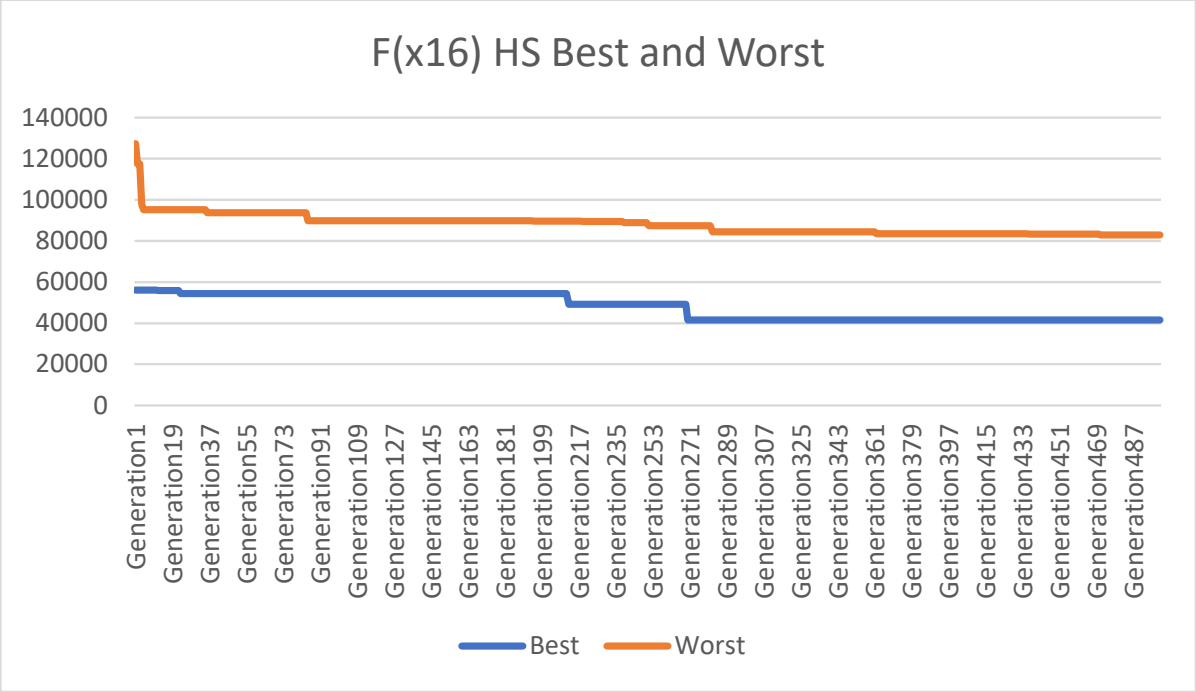


Figure 3.51: HS Algorithm with Step Function

Function17.pdf

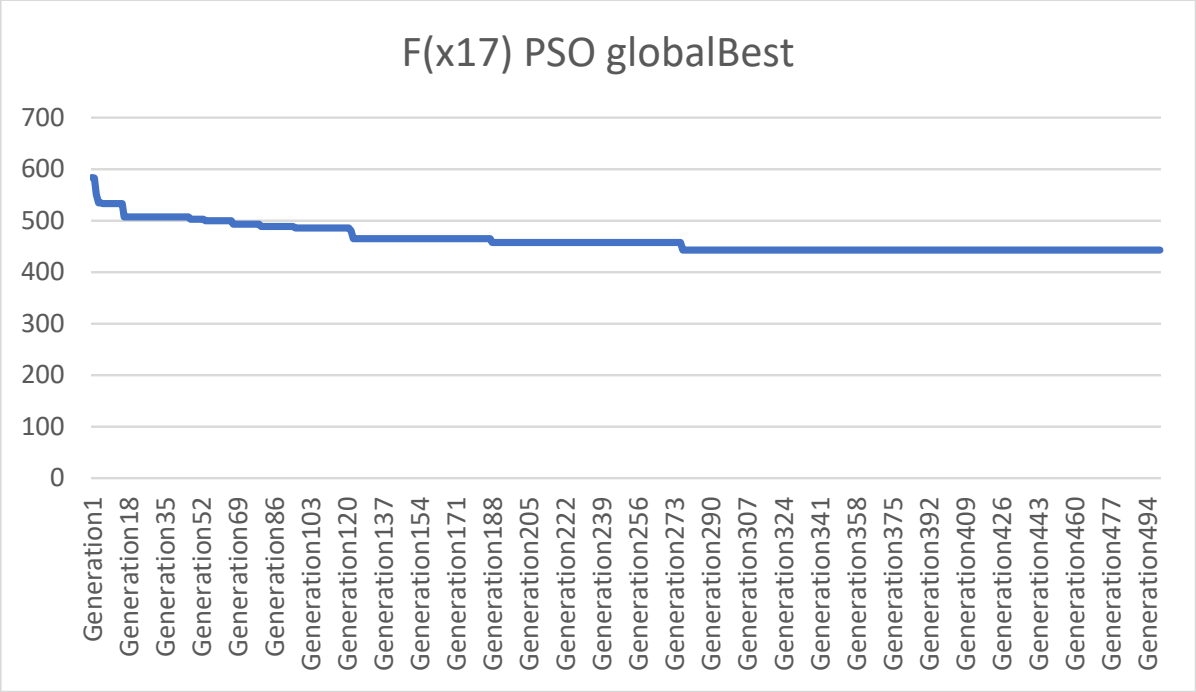


Figure 3.52: PSO Algorithm with Alpine Function

Function17.pdf

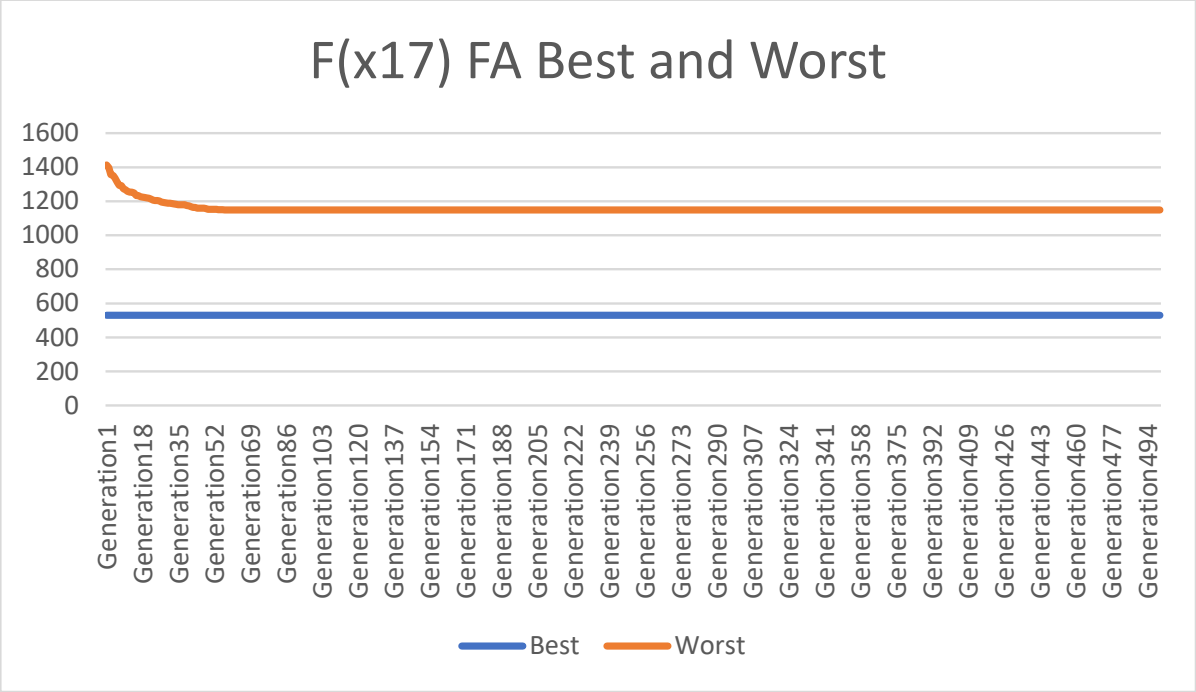


Figure 3.53: FA Algorithm with Alpine Function

Function17.pdf

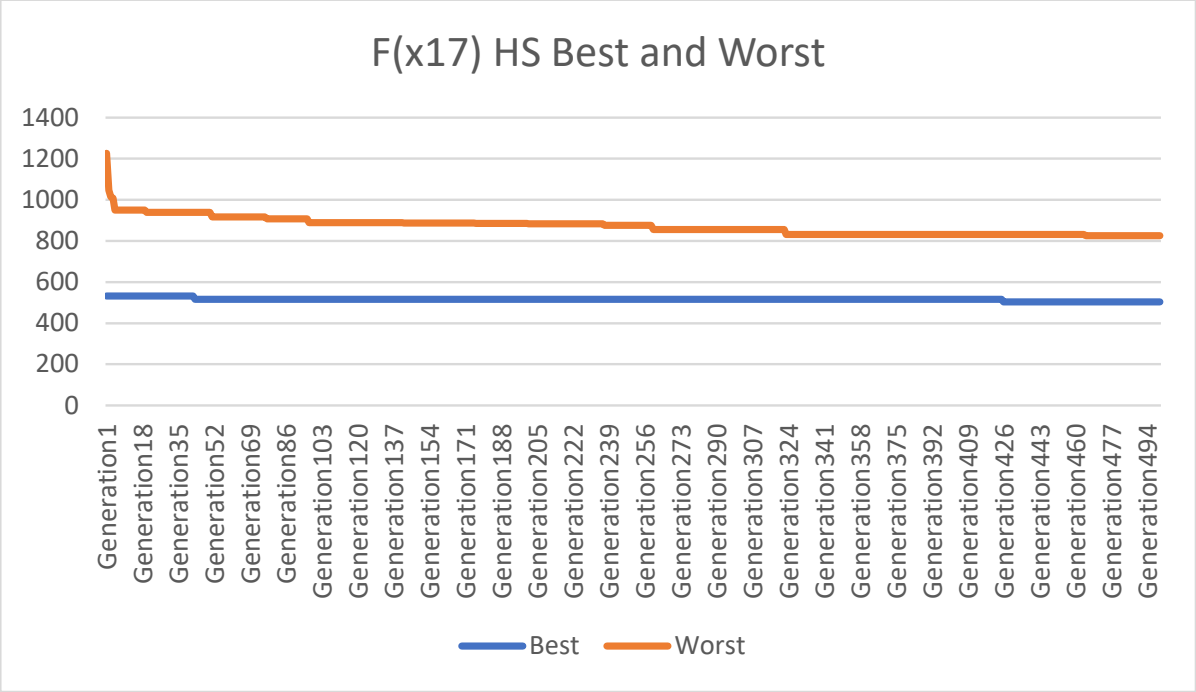
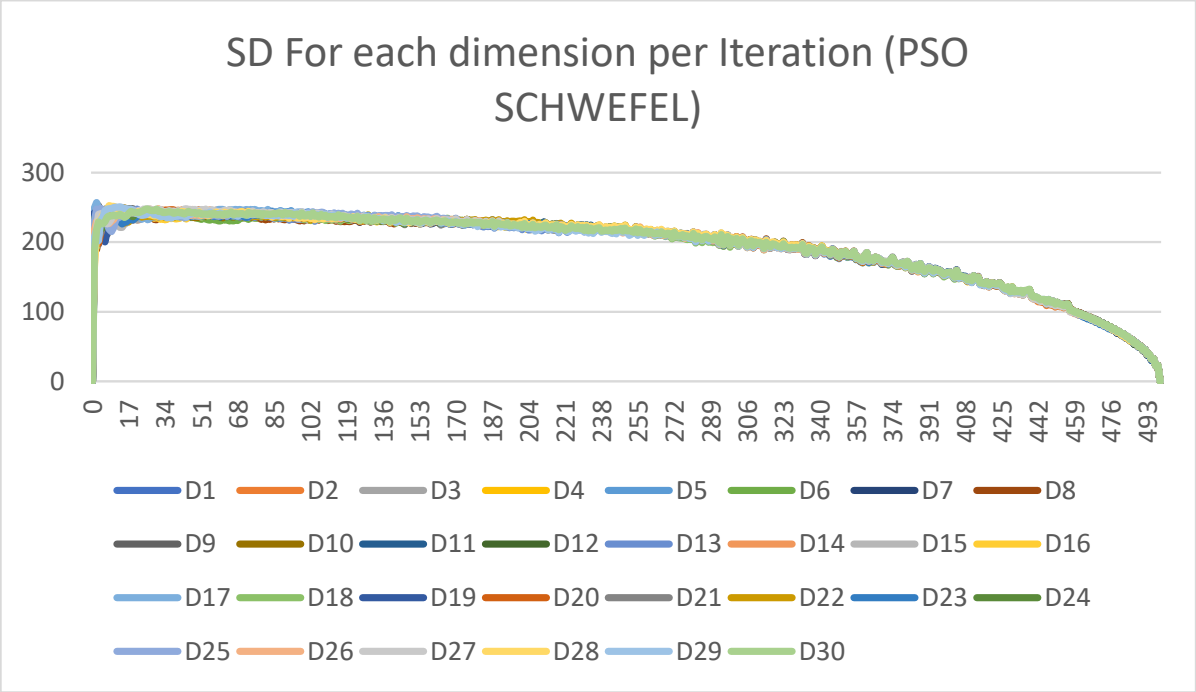
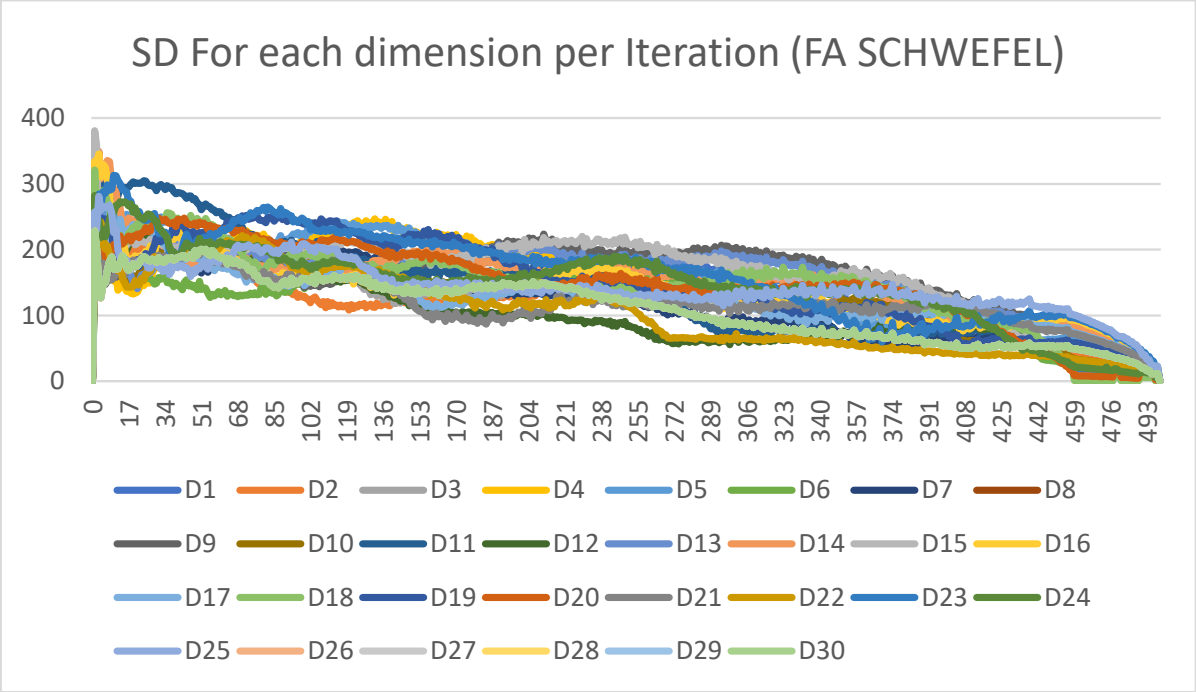


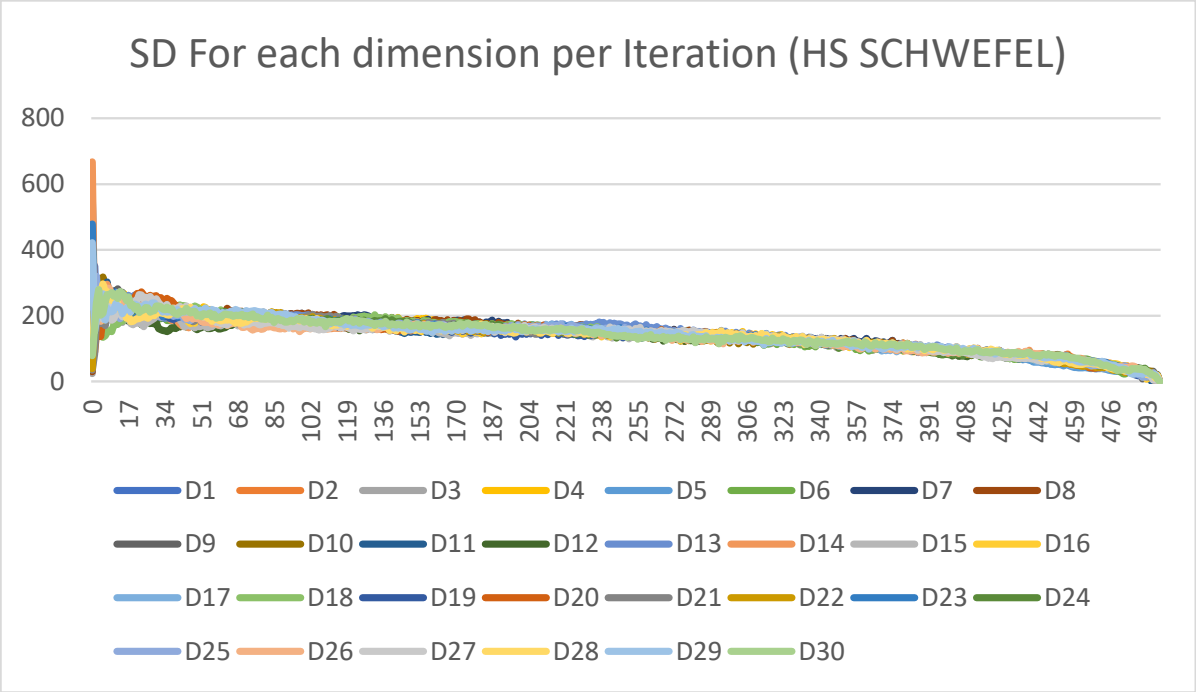
Figure 3.54: HS Algorithm with Alpine Function

STAG.pdf


SD For each dimension per Iteration (PSO SCHWEFEL)
300
200
100
0
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30

STAG.pdf


SD For each dimension per Iteration (FA SCHWEFEL)
400
300
200
100
0
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30


SD For each dimension per Iteration (HS SCHWEFEL)
800
600
400
200
0
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30