Matthew Harker

CS 471

Project 4

An Analysis of Data Passed through the Particle Swarm, Firefly, and Harmony Search Optimization Algorithms

Central Washington University
Department of Computer Science

May 20, 2019

1 Intro

This project deal with the topic of three methods of generating optimal solution vectors for various functions. The three methods being used are the Particle Swarm Algorithm, Firefly Algorithm, and Harmony Search algorithm. All three of these algorithms work similarly, by having a swarm of solution vectors and allowing those to be mutated by other solutions in the population. This influence of other solutions helps increase the randomness of the algorithms, as having randomness increases the overall search space which helps prevent stagnation, as well as increasing the area that is covered.

For each algorithm, the parameters of the populations themselves were kept the same. Each solution vector had thirty dimensions, each population contained five hundred solution vectors, and the populations underwent 500 iterations of optimization. There are also multiple values that modify the way each algorithm acts. For Particle Swarm the three values were a velocity dampener "k", and two velocity constants "c1" and "c2". c1 affects the personal best solution vector, and c2 affects the global best solution vector. For this experiment, k was set to 0.05, c1 was set to 0.05, and c2 was set to 0.05. Firefly also had three values: a movement constant "alpha", an attractiveness modifier "beta", and another movement constant "gamma". These values were set to 0.05, 0.8, and 0.001 respectively. Lastly, Harmony Search also had three constants: Bandwidth, Harmony Memory Considering Rate (HMCR), and Pitch Adjusting Ratio (PAR).

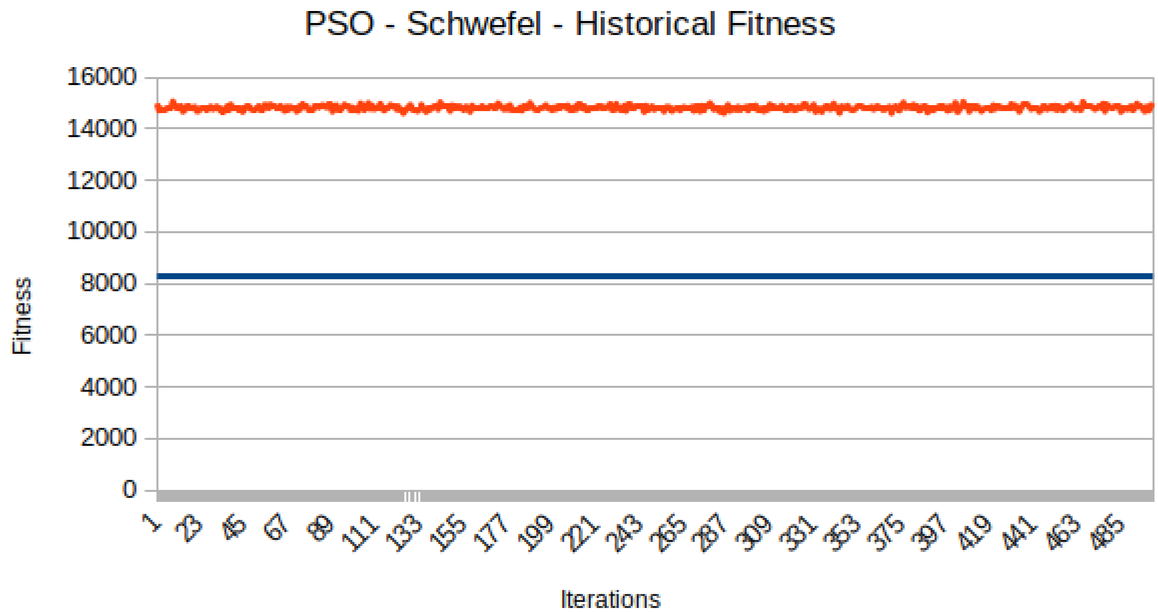
To prevent any potential confusion, some clarifications will be made. "Function calls" refers to how many times the base functions (Schwefel, Rastrigin, etc) have been ran. "Average" is the mean of all the resulting data, "STDev" is the standard deviation, and "Median" is the piece of data that is in the center of the data set. Range refers to the difference between the maximum and the minimum produced values of each vector. Time refers to how long it took each optimization algorithm to run an iteration of the algorithm in milliseconds (unless otherwise stated). Each algorithm has elected to use a different thematic name for a solution vector. For Particle Swarm algorithm, they are called "particles", for Firefly algorithm they are called "fireflies", and for Harmony Search algorithm they are known as "harmonies".

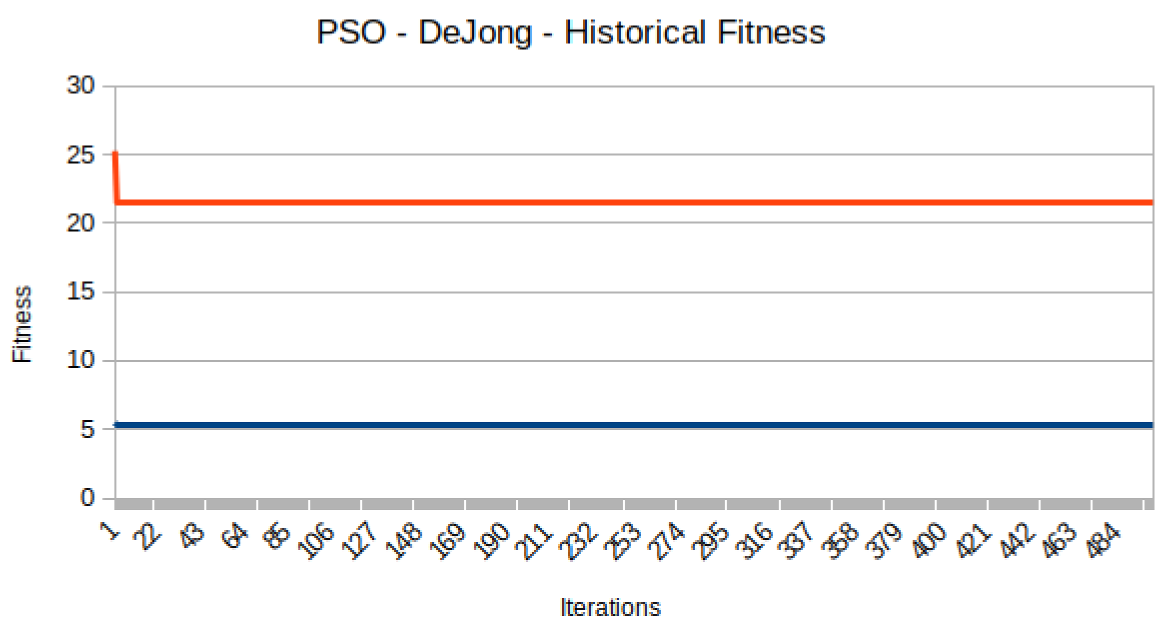
2 Fitness

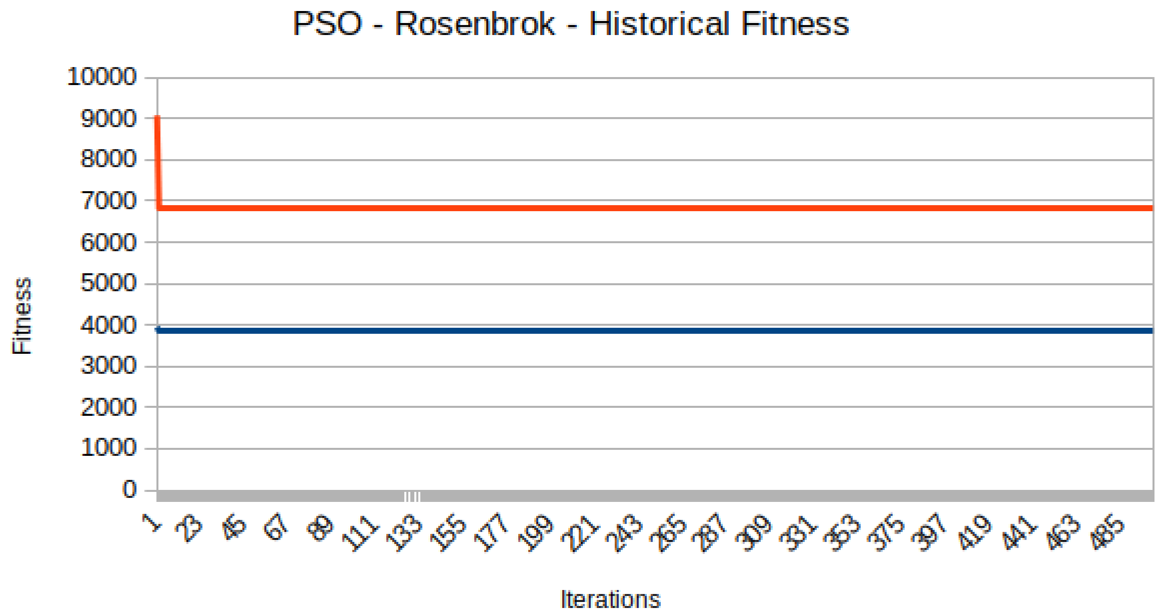
The whole idea of these optimization algorithms is to progressively create smaller and smaller fitness values for each function, until they are either at or near the optimal value. For this test, the optimal value is assumed to be the lowest value possible, which includes negative values. The following graphs and tables will present the data about the resulting final optimized fitnesses of each algorithm and function, as well as their changes over time.

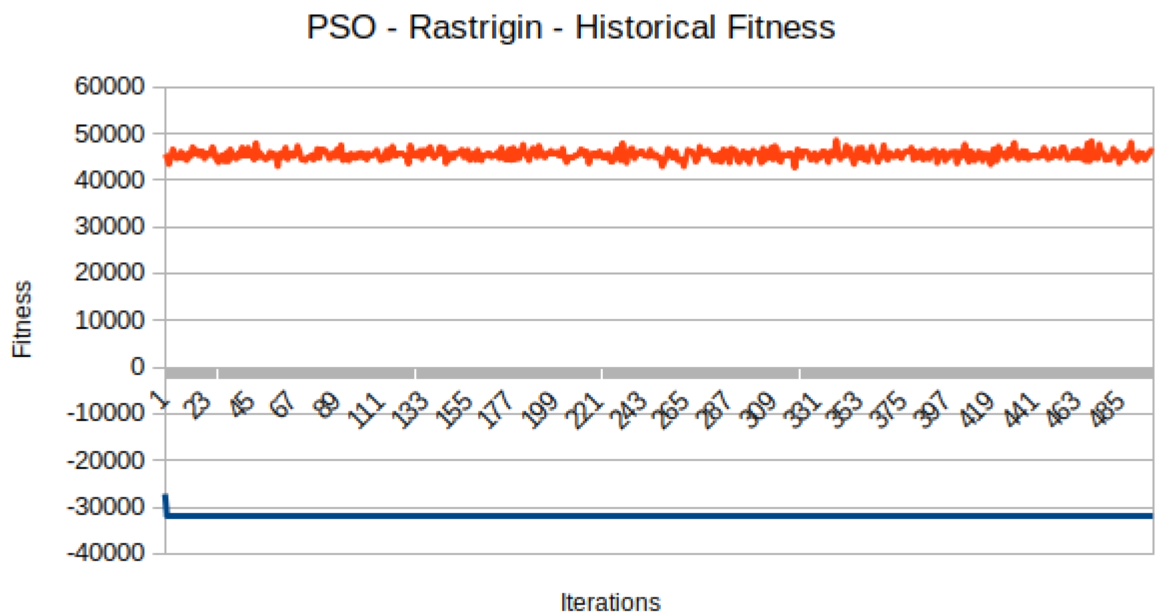
The first series of figures are going to be from the Particle Swarm algorithm. This group of figures is going to not include the change in the worst fitness, but instead will show off the change in the personal best fitness of the particles.

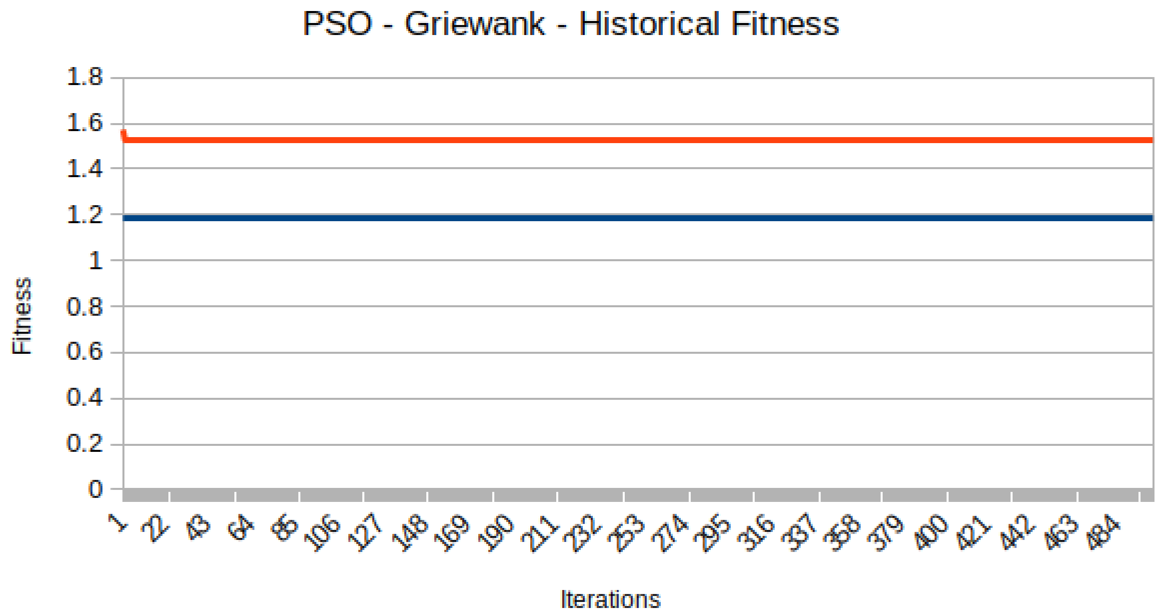
For the following graphs, the red line indicates the historical worst fitness of the population, whereas the blue line indicates the historical best fitness of the population.


PSO - Schwefel - Historical Fitness
Fitness
Iterations
16000
14000
12000
10000
8000
6000
4000
2000
0
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485

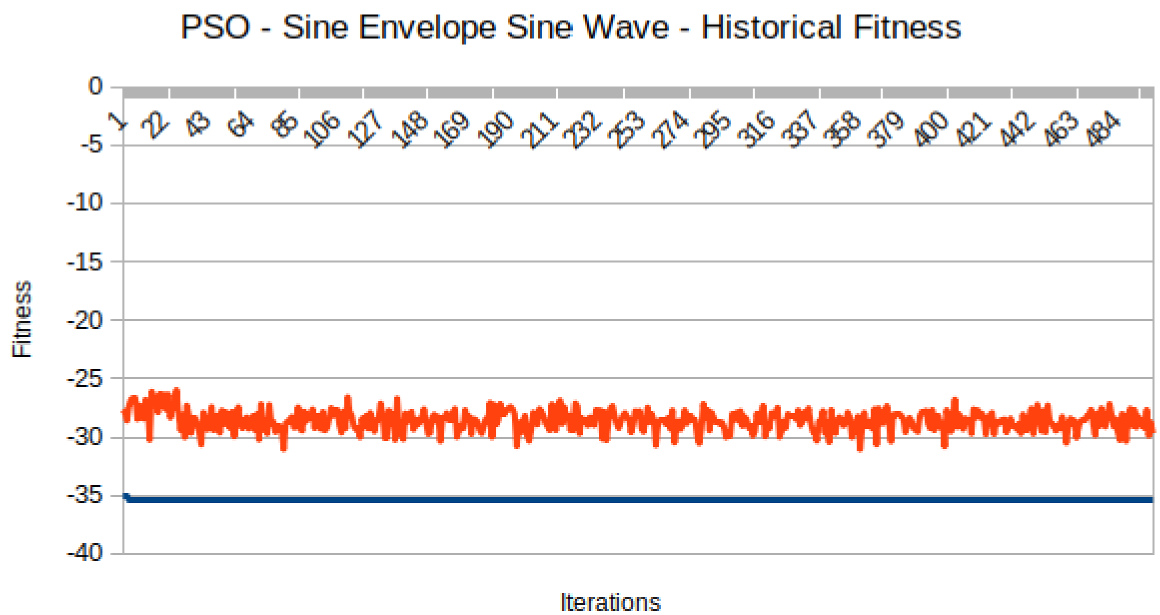

PSO - DeJong - Historical Fitness
Fitness
Iterations
30
25
20
15
10
5
0
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484


PSO - Rosenbrok - Historical Fitness
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

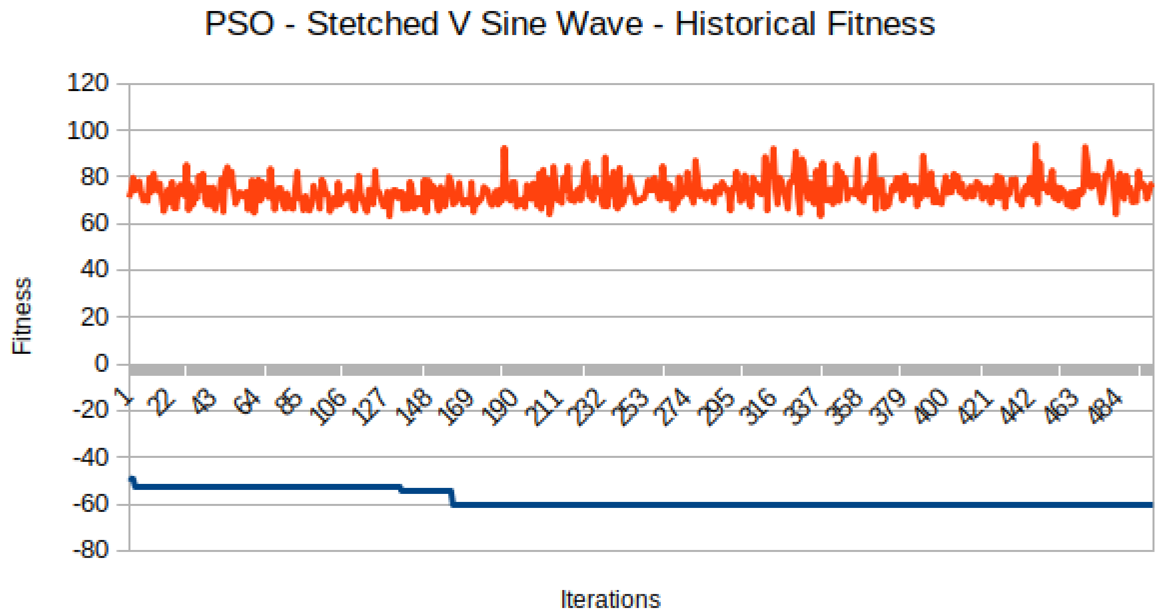

PSO - Rastrigin - Historical Fitness
60000
50000
40000
30000
20000
10000
0
-10000
-20000
-30000
-40000
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

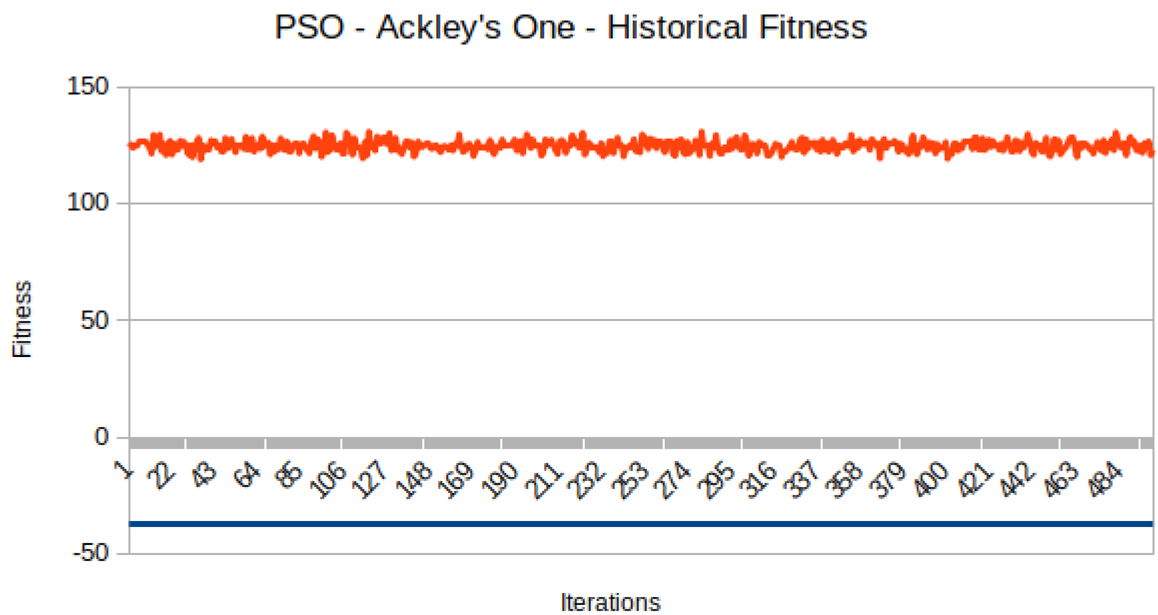


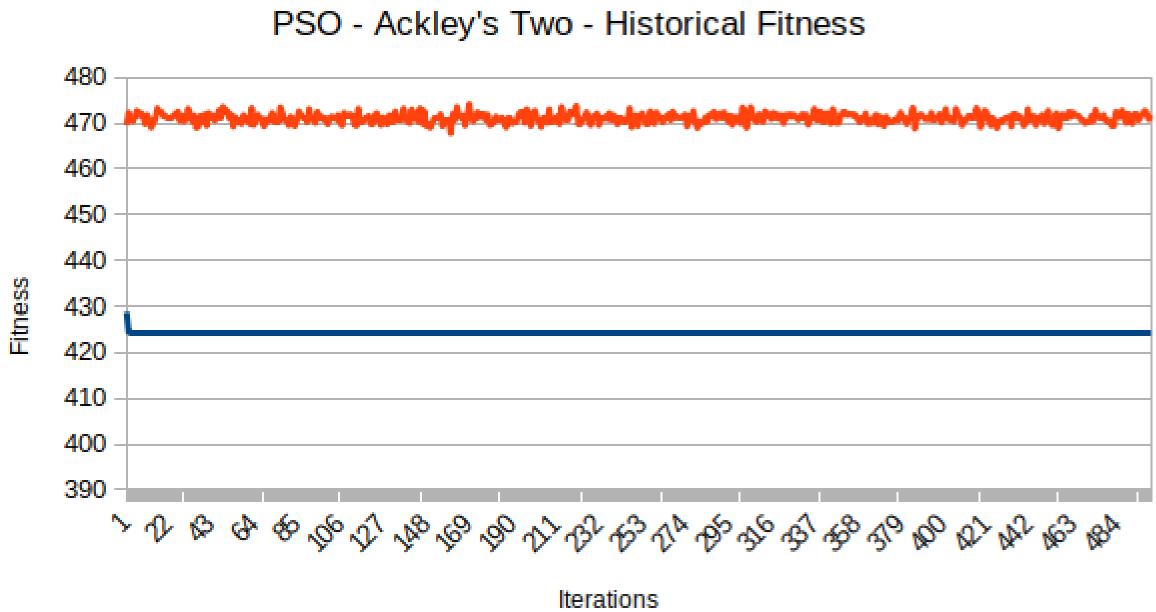
PSO - Griewank - Historical Fitness
Fitness
Iterations

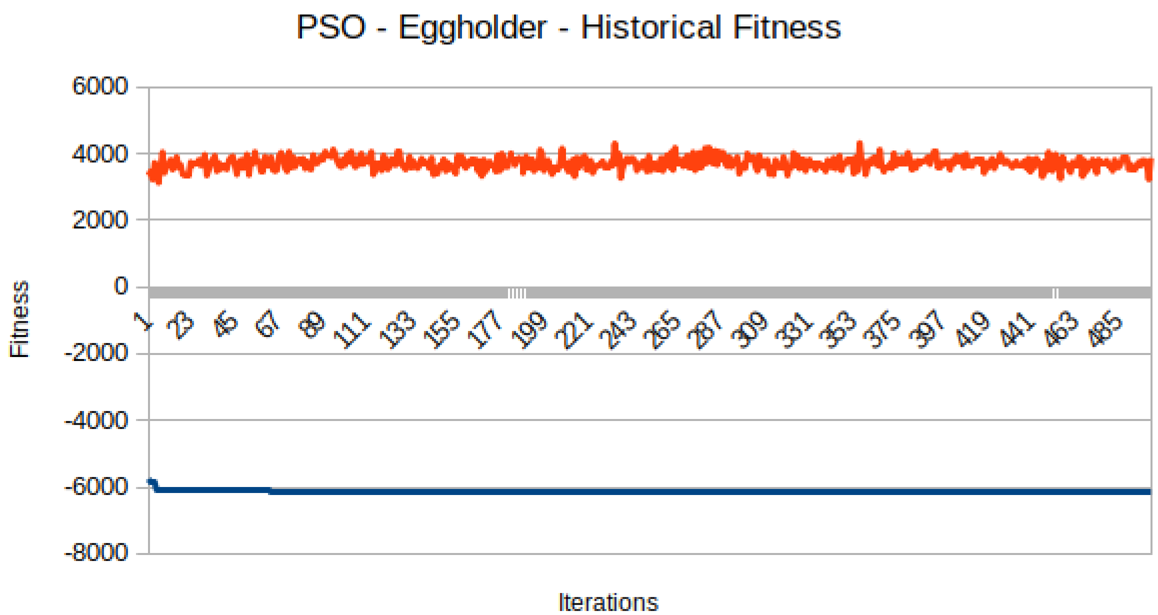


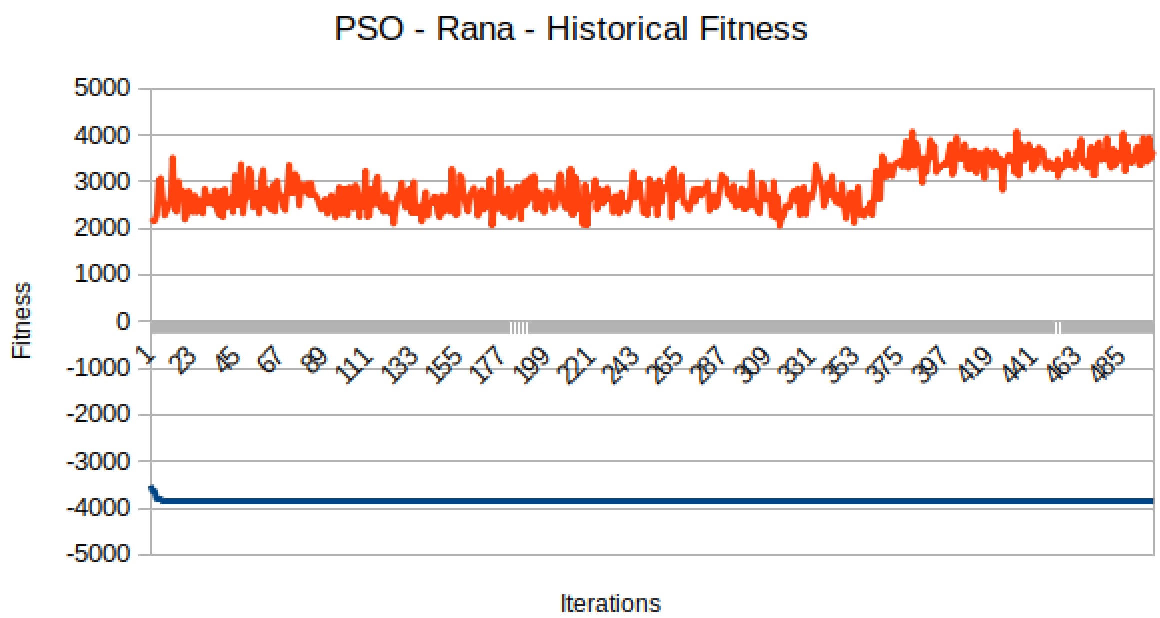
PSO - Sine Envelope Sine Wave - Historical Fitness
Fitness
Iterations

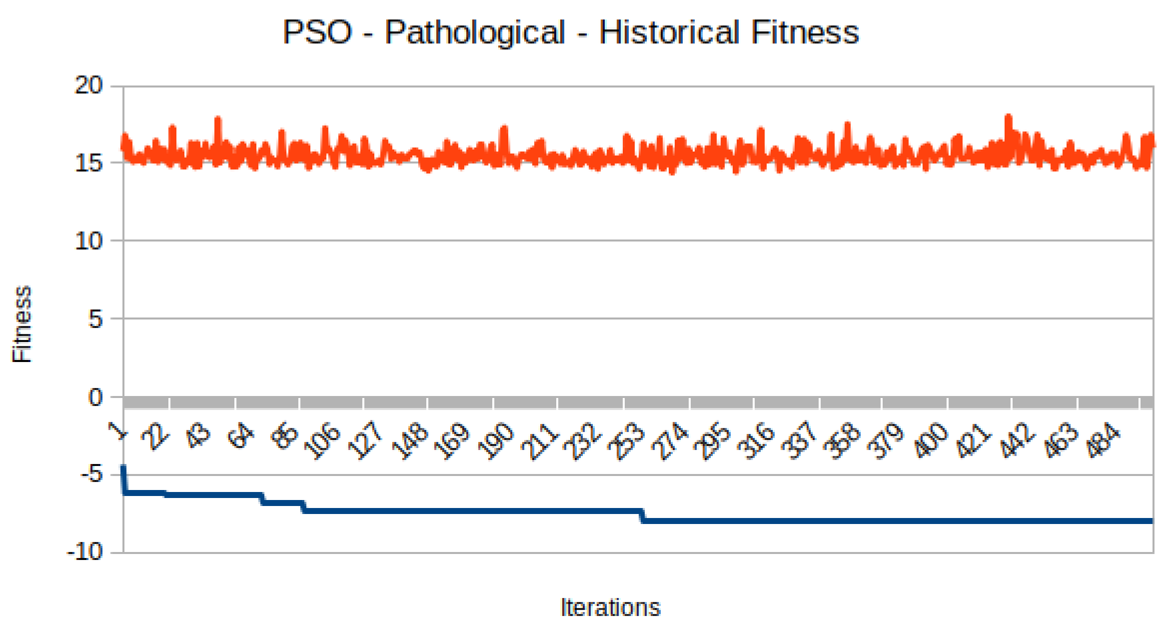

PSO - Stetched V Sine Wave - Historical Fitness
Fitness
Iterations
120
100
80
60
40
20
0
-20
-40
-60
-80
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

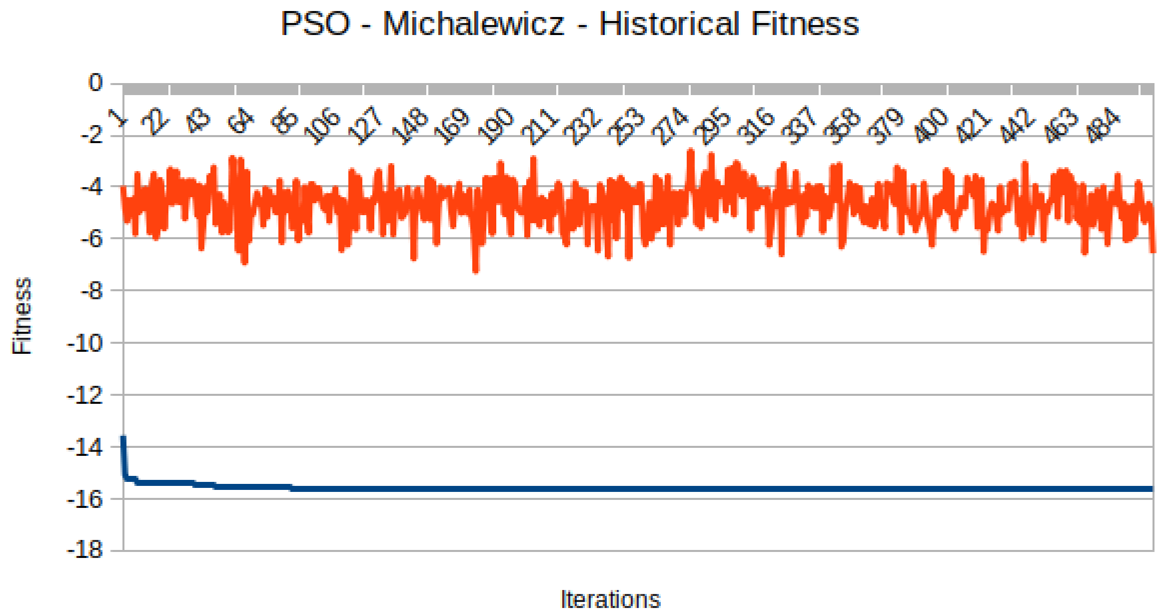

PSO - Ackley's One - Historical Fitness
Fitness
Iterations
150
100
50
0
-50
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

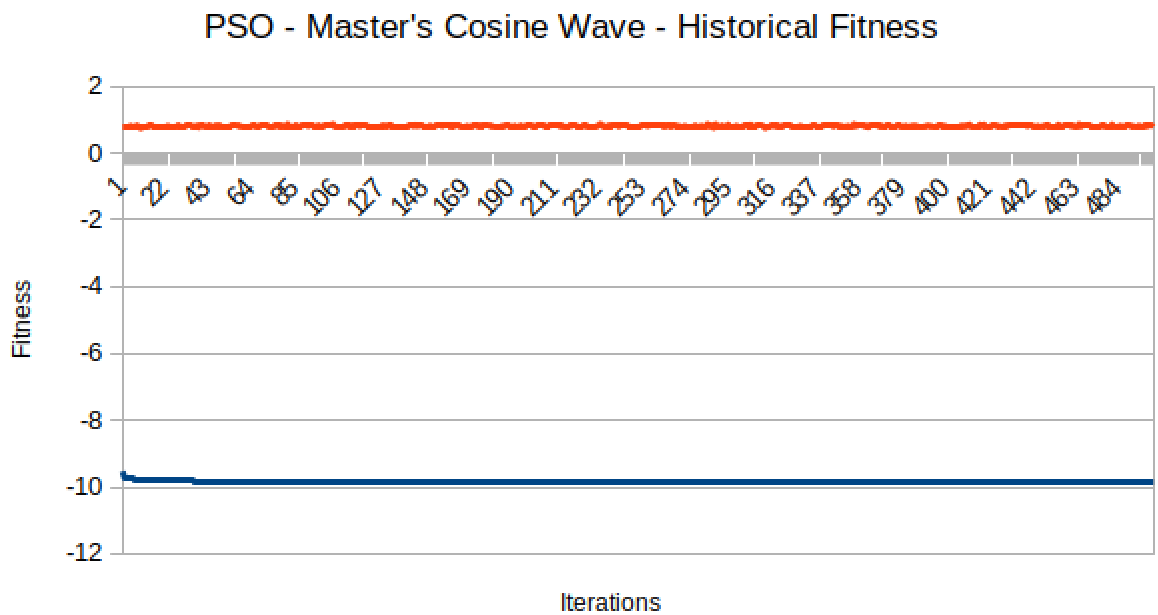

PSO - Ackley's Two - Historical Fitness
480
470
460
450
440
430
420
410
400
390
Fitness
1 22 43 64 85 106 127 148 169 190 211 232 253 274 295 316 337 358 379 400 421 442 463 484
Iterations

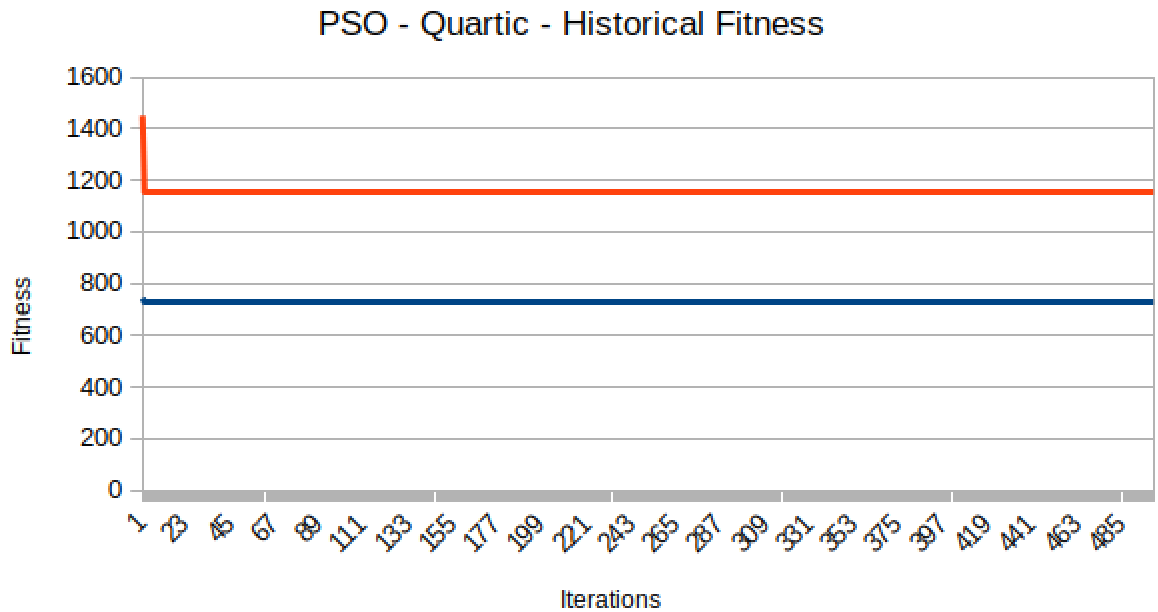

PSO - Eggholder - Historical Fitness
6000
4000
2000
0
-2000
-4000
-6000
-8000
Fitness
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485
Iterations

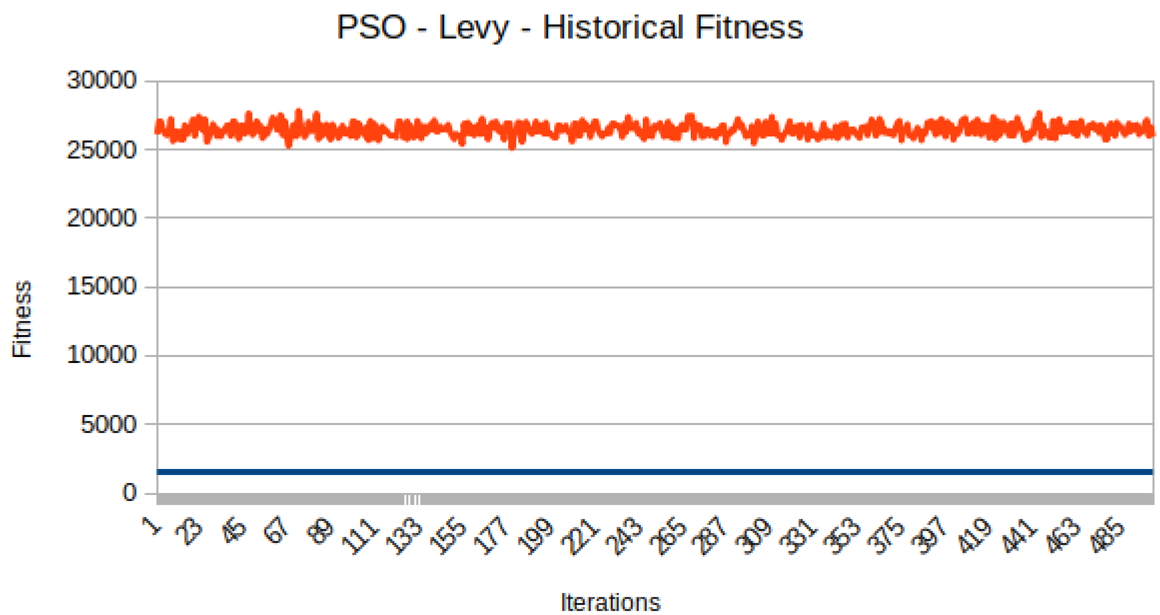

PSO - Rana - Historical Fitness
Fitness
Iterations
5000
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
-5000
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485

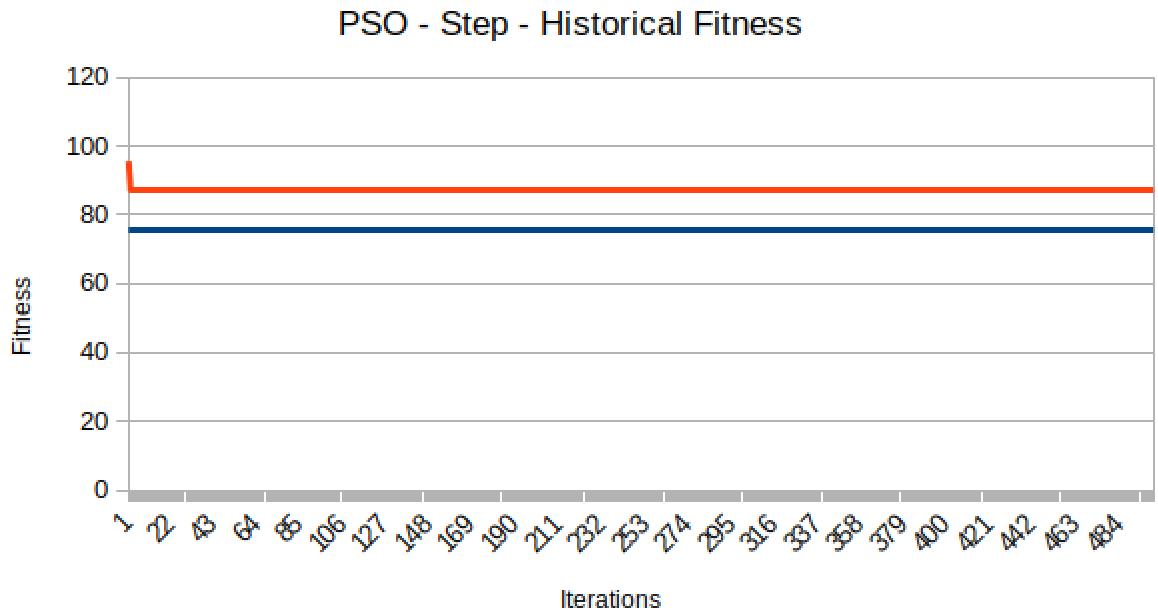

PSO - Pathological - Historical Fitness
Fitness
Iterations
20
15
10
5
0
-5
-10
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

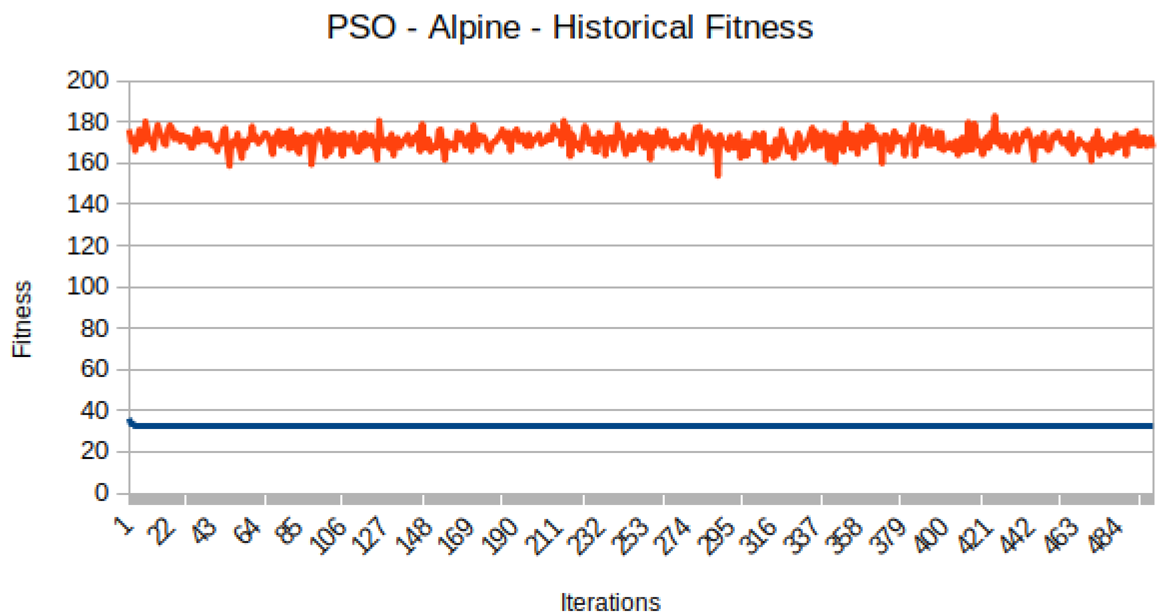

PSO - Michalewicz - Historical Fitness
Fitness
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
1 22 43 64 85 106 127 148 169 190 211 232 253 274 295 316 337 358 379 400 421 442 463 484
Iterations

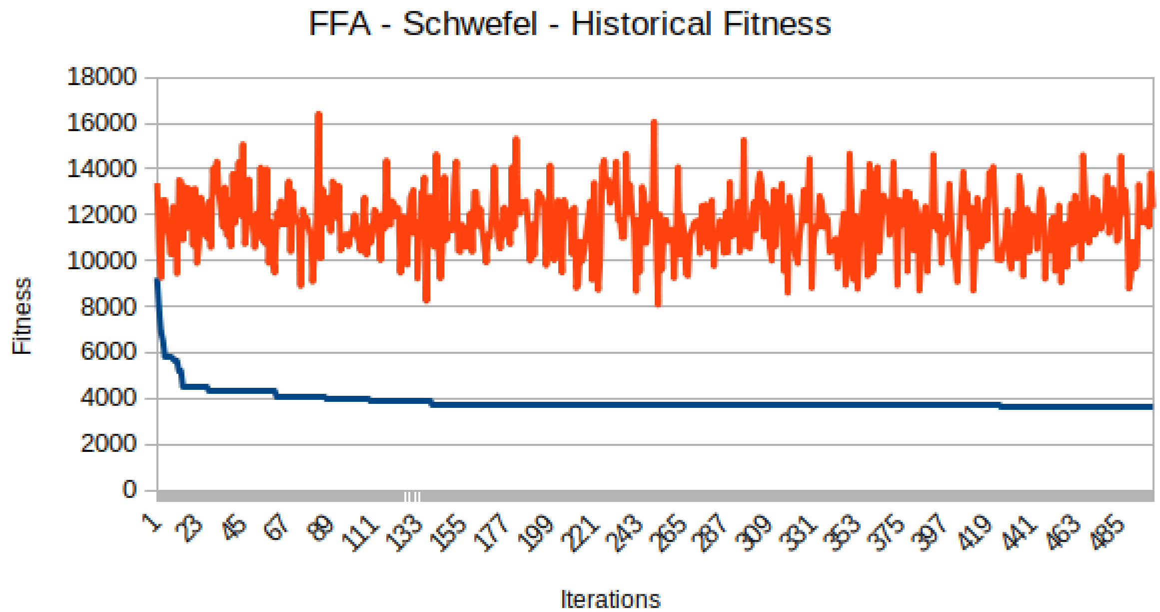

PSO - Master's Cosine Wave - Historical Fitness
Fitness
2
0
-2
-4
-6
-8
-10
-12
1 22 43 64 85 106 127 148 169 190 211 232 253 274 295 316 337 358 379 400 421 442 463 484
Iterations

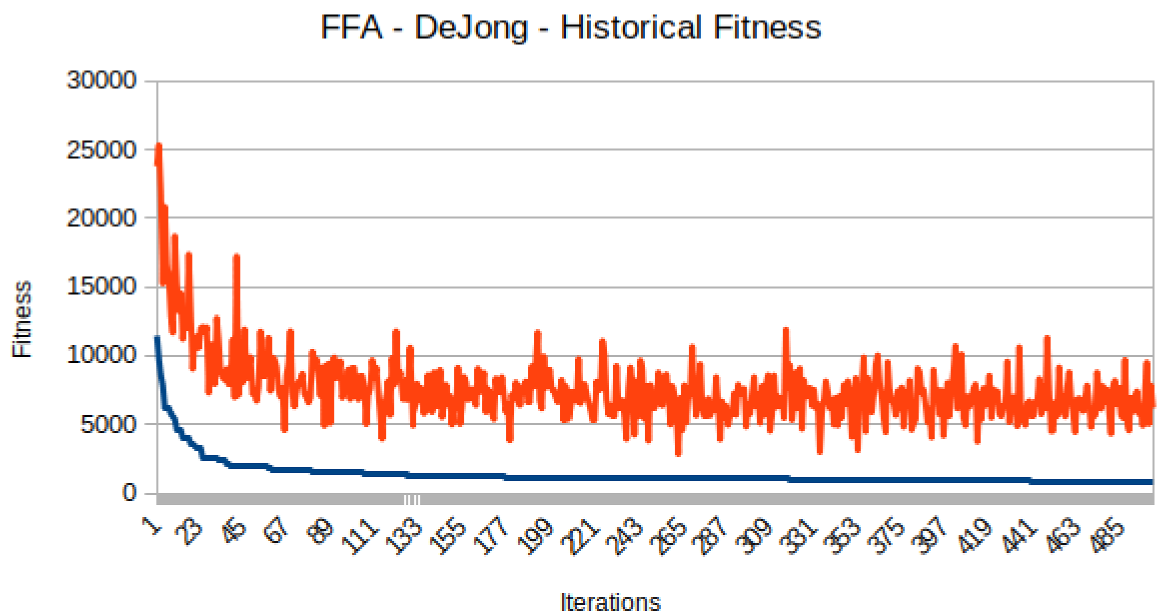

PSO - Quartic - Historical Fitness
1600
1400
1200
1000
800
600
400
200
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

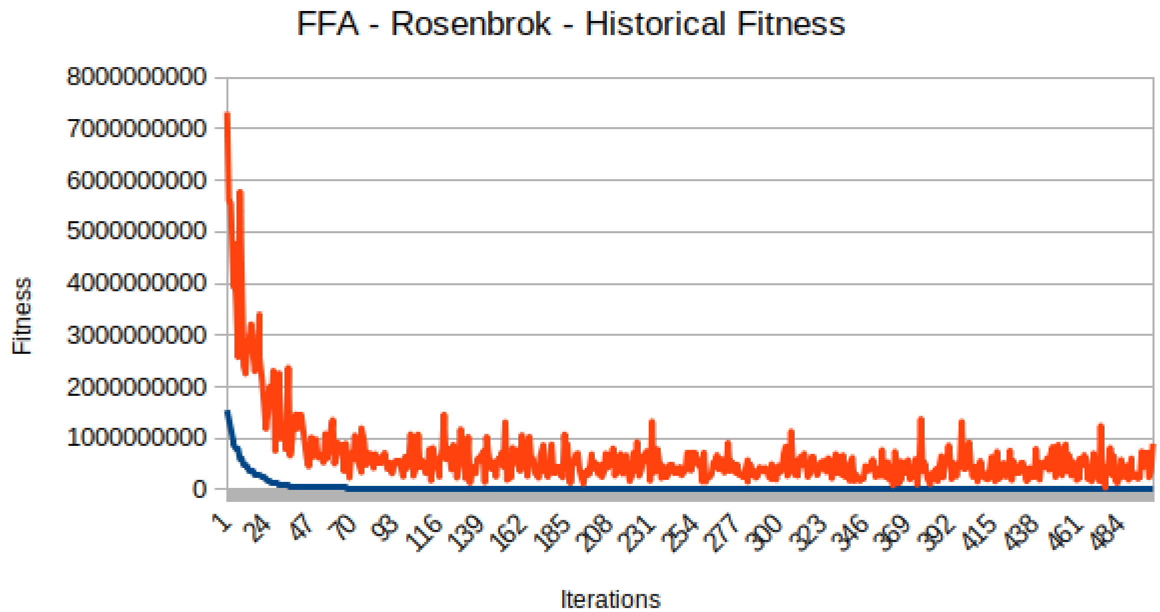

PSO - Levy - Historical Fitness
30000
25000
20000
15000
10000
5000
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

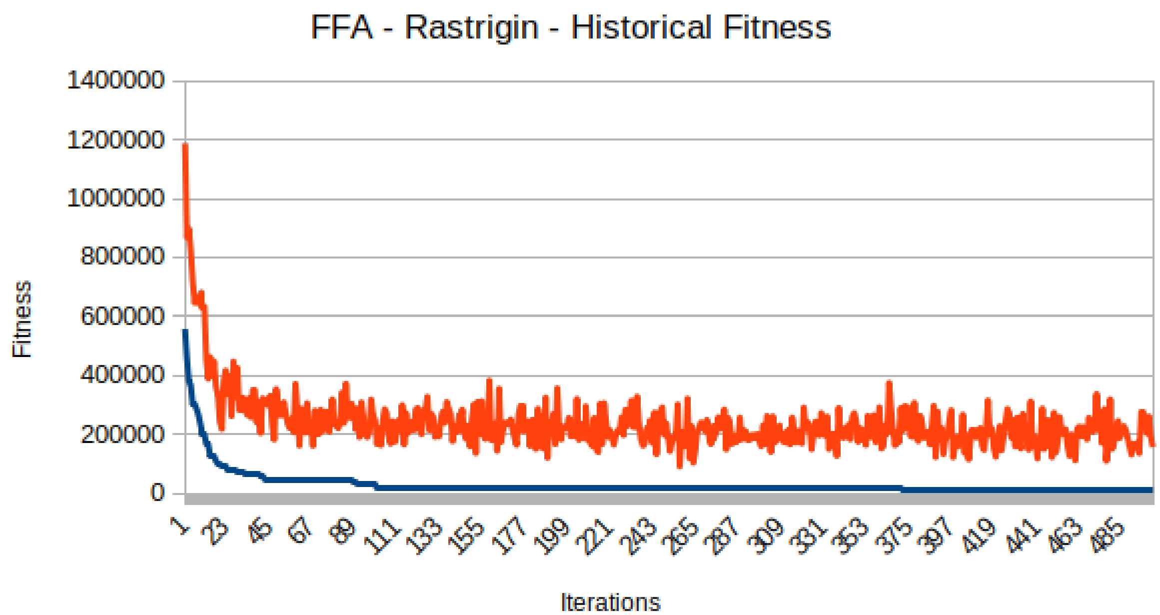

PSO - Step - Historical Fitness
120
100
80
60
40
20
0
Fitness
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484
Iterations

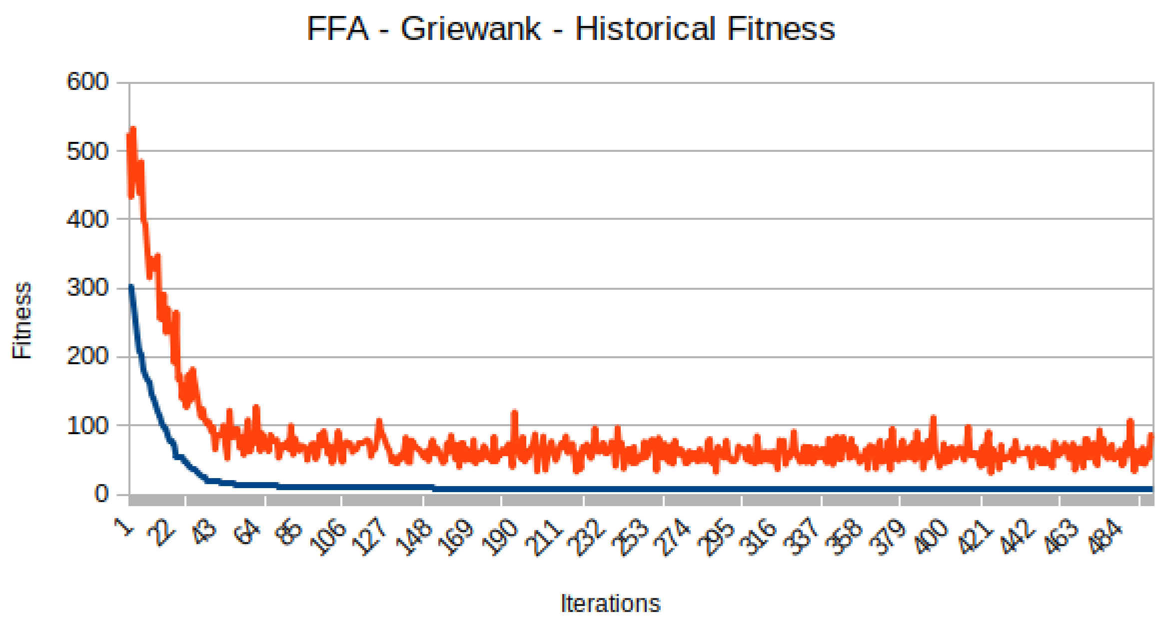

PSO - Alpine - Historical Fitness
200
180
160
140
120
100
80
60
40
20
0
Fitness
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484
Iterations

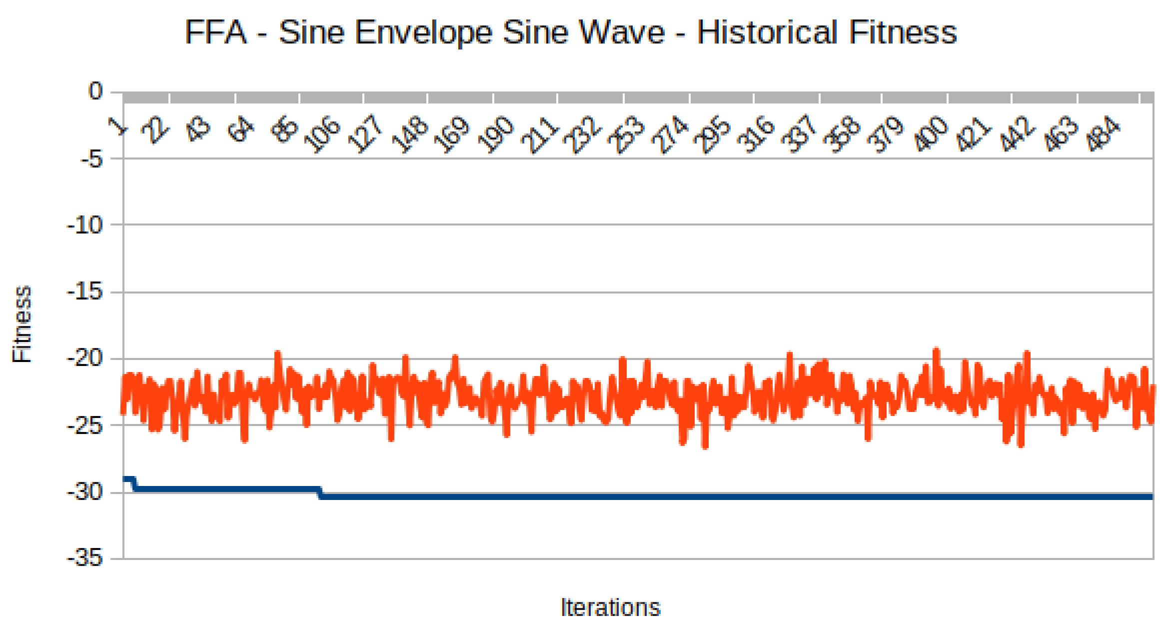

FFA - Schwefel - Historical Fitness
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

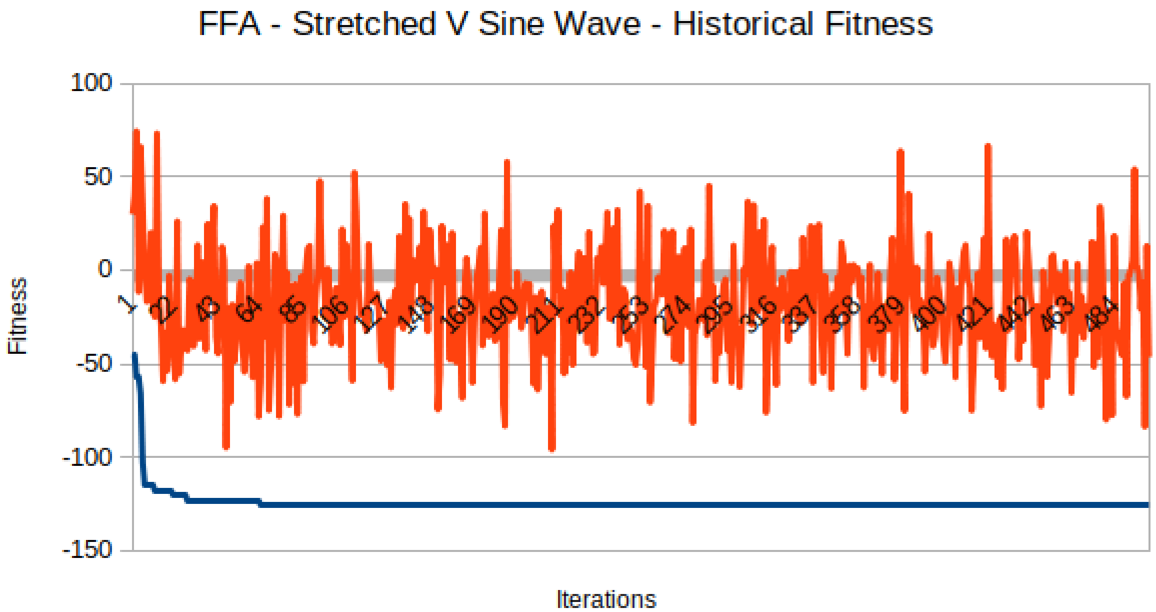

FFA - DeJong - Historical Fitness
30000
25000
20000
15000
10000
5000
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

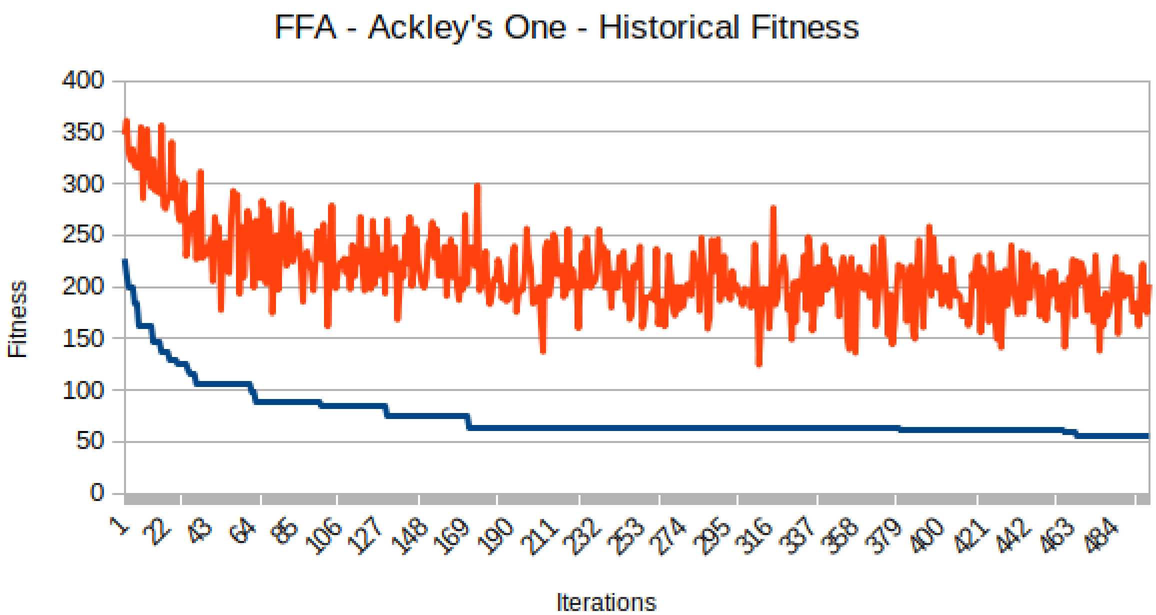

FFA - Rosenbrok - Historical Fitness
8000000000
7000000000
6000000000
5000000000
4000000000
3000000000
2000000000
1000000000
0
Fitness
1 24 47 70 93 116 139 162 185 208 231 254 277 300 323 346 369 392 415 438 461 484
Iterations

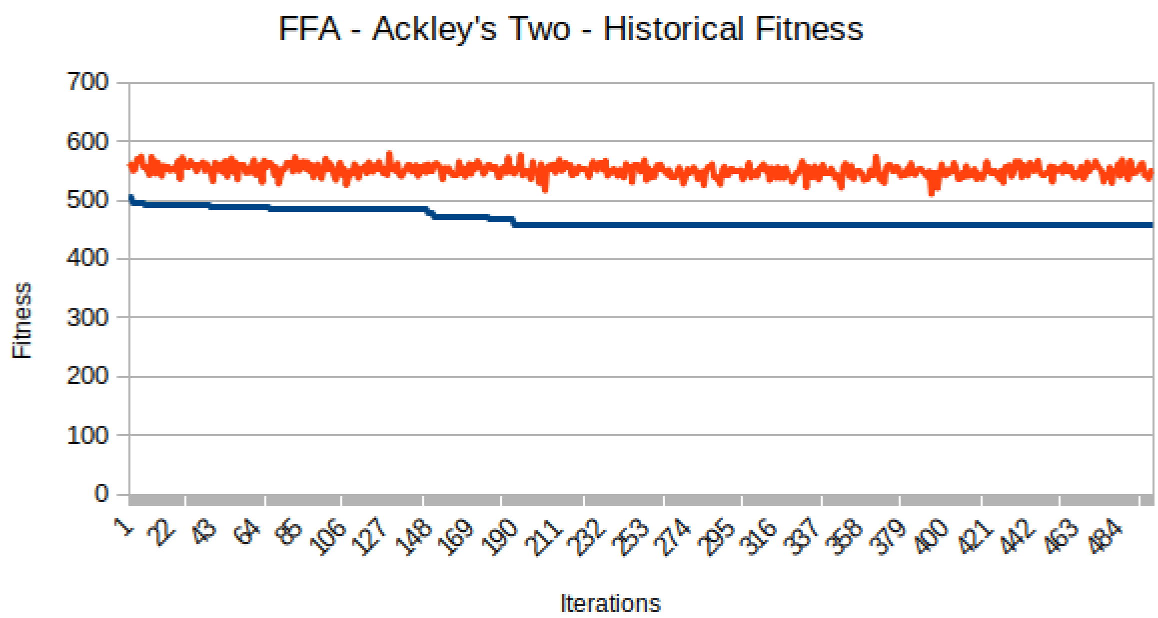

FFA - Rastrigin - Historical Fitness
1400000
1200000
1000000
800000
600000
400000
200000
0
Fitness
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485
Iterations

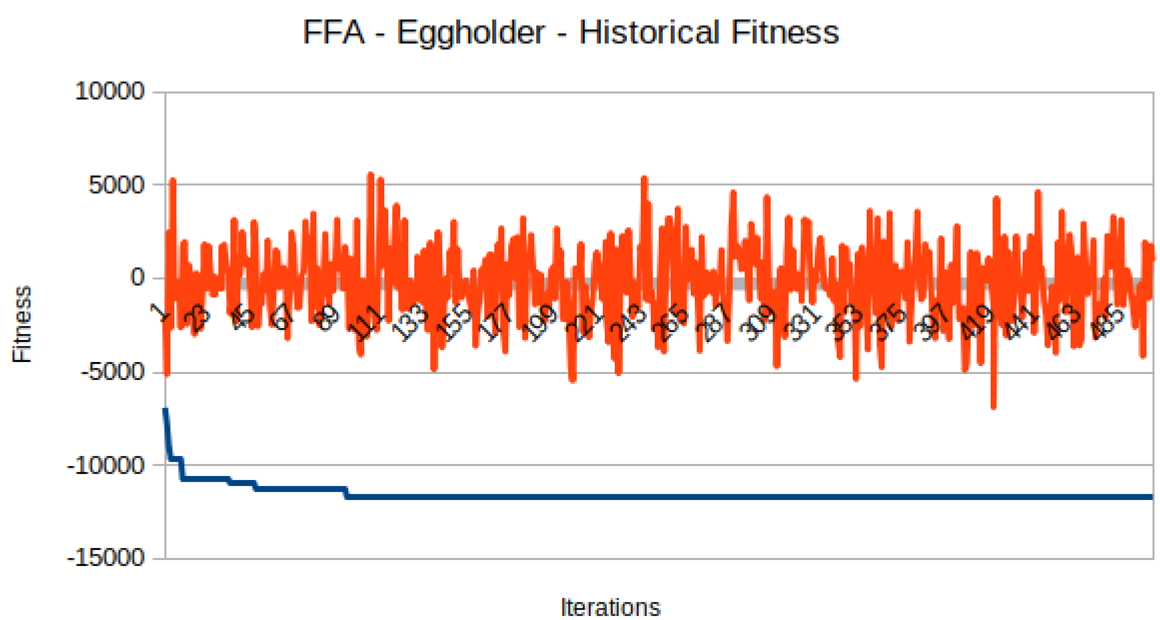

FFA - Griewank - Historical Fitness
Fitness
Iterations
600
500
400
300
200
100
0
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484


FFA - Sine Envelope Sine Wave - Historical Fitness
Fitness
Iterations
0
-5
-10
-15
-20
-25
-30
-35
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

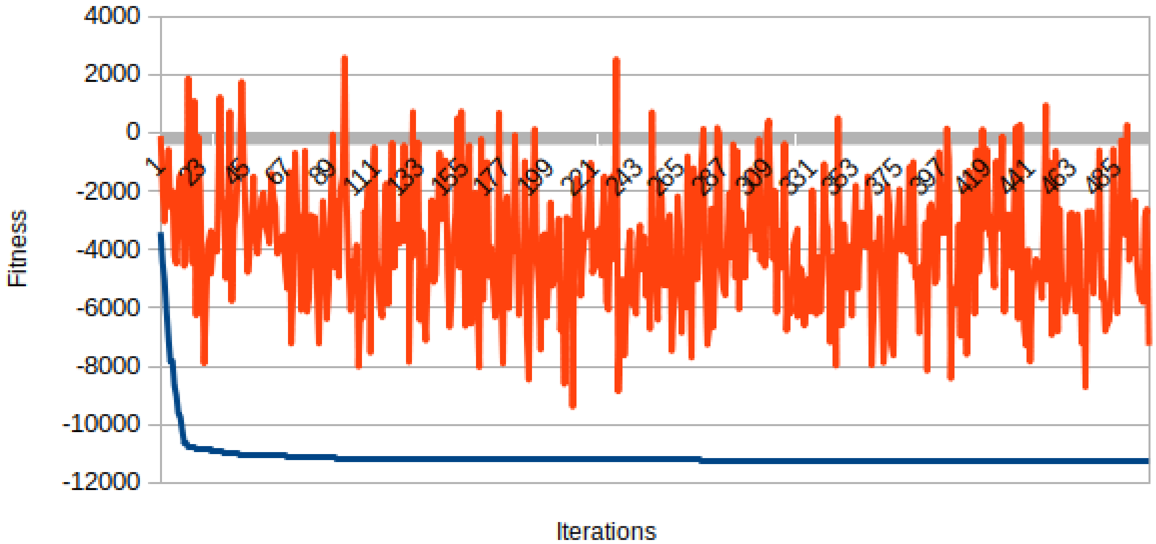

FFA - Stretched V Sine Wave - Historical Fitness
100
50
0
-50
-100
-150
Fitness
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484
Iterations


FFA - Ackley's One - Historical Fitness
400
350
300
250
200
150
100
50
0
Fitness
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484
Iterations

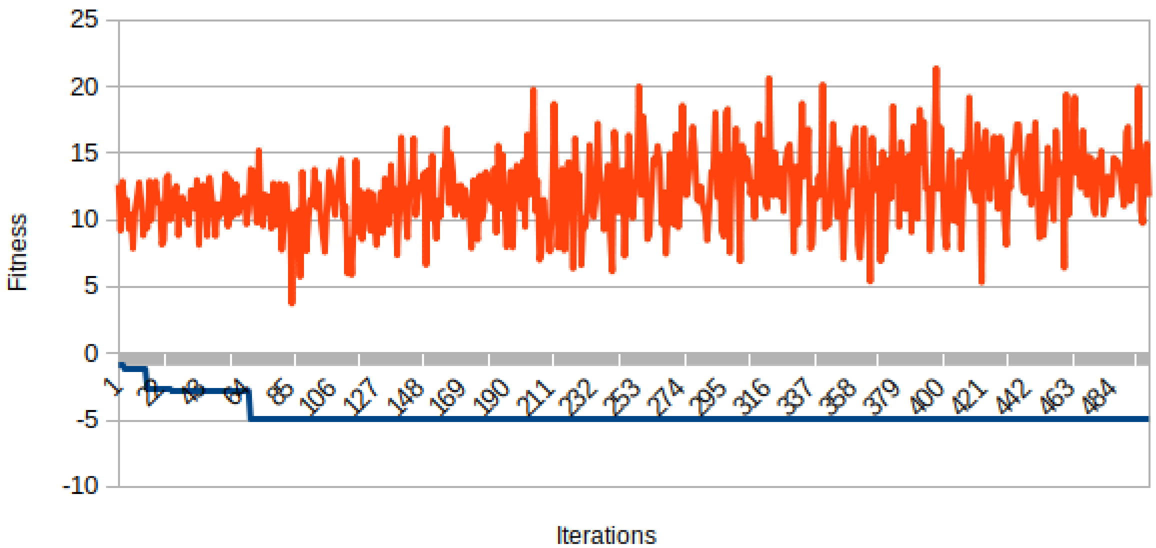

FFA - Ackley's Two - Historical Fitness
700
600
500
400
300
200
100
0
Fitness
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484
Iterations


FFA - Eggholder - Historical Fitness
10000
5000
0
-5000
-10000
-15000
Fitness
Iterations

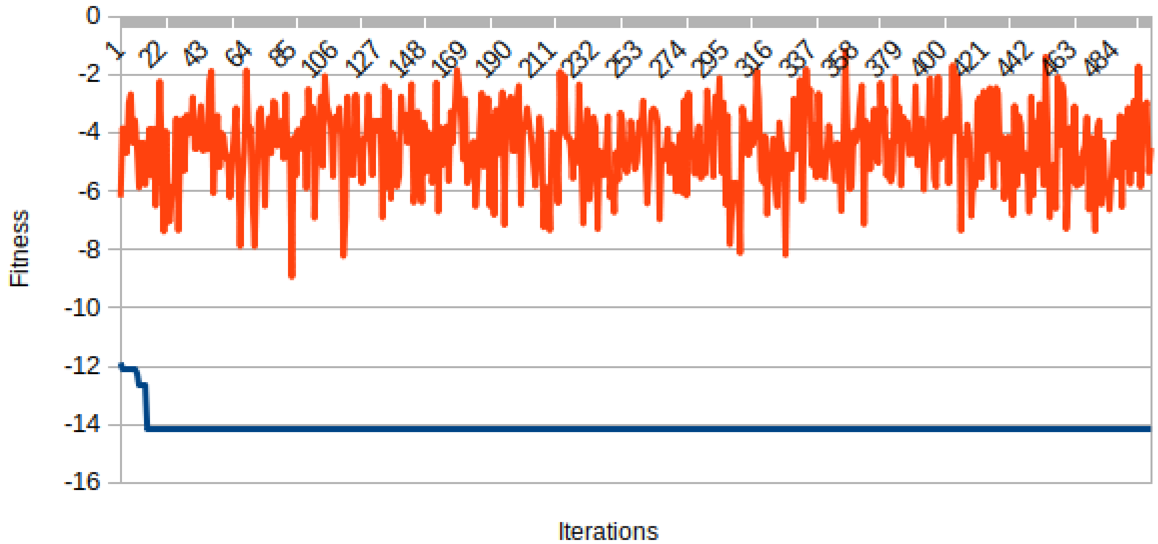
FFA - Rana - Historical Fitness



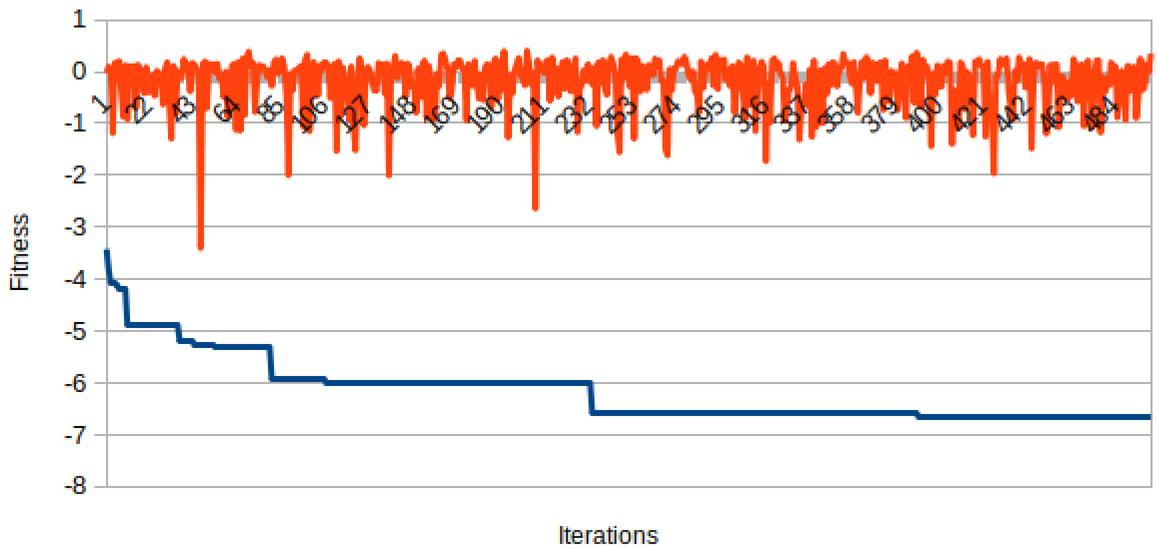
FFA - Pathological - Historical Fitness

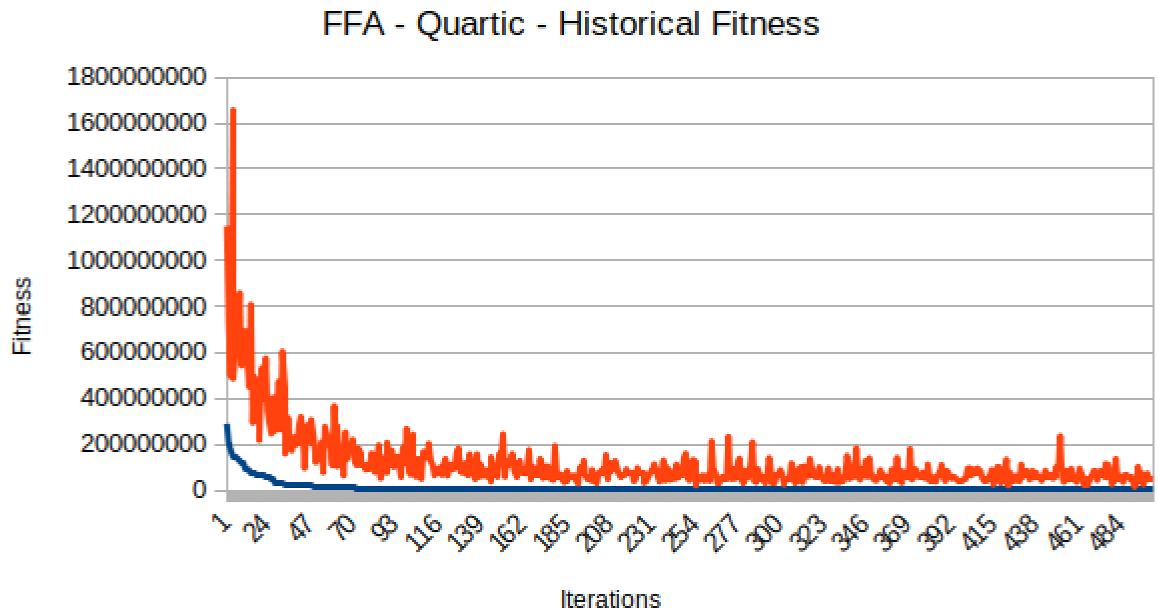


FFA - Michalewicz - Historical Fitness

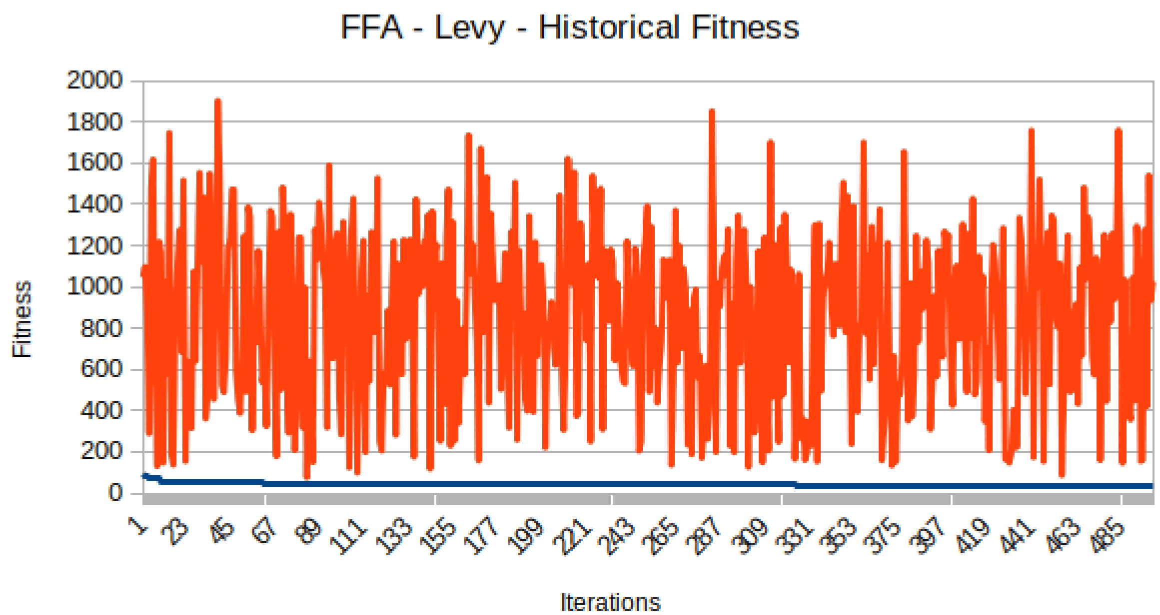


FFA - Master's Cosine Wave - Historical Fitness

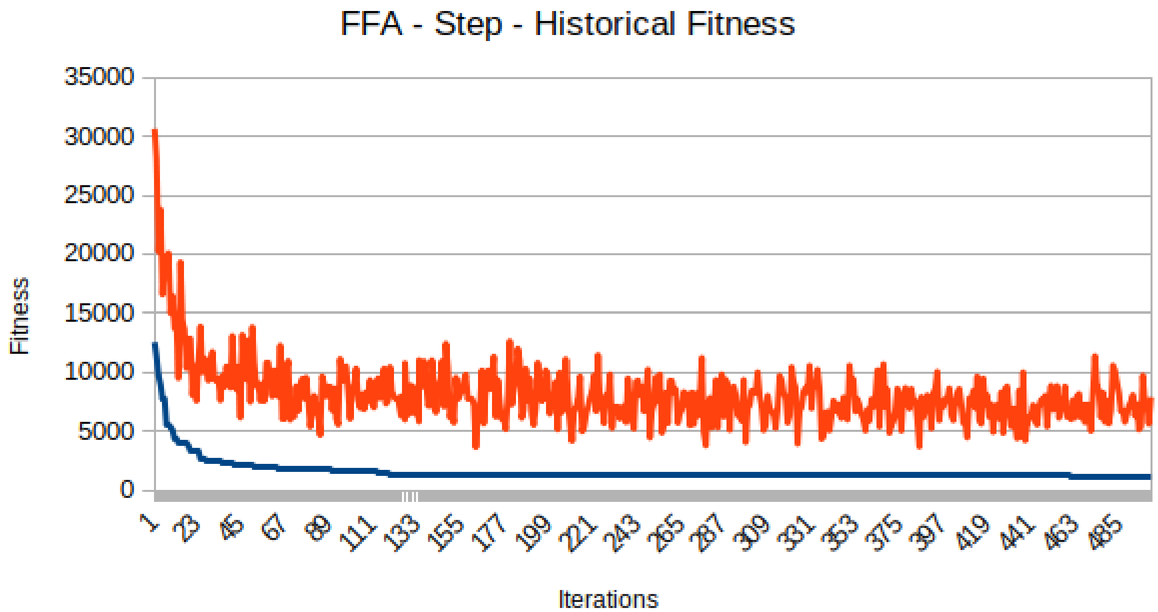


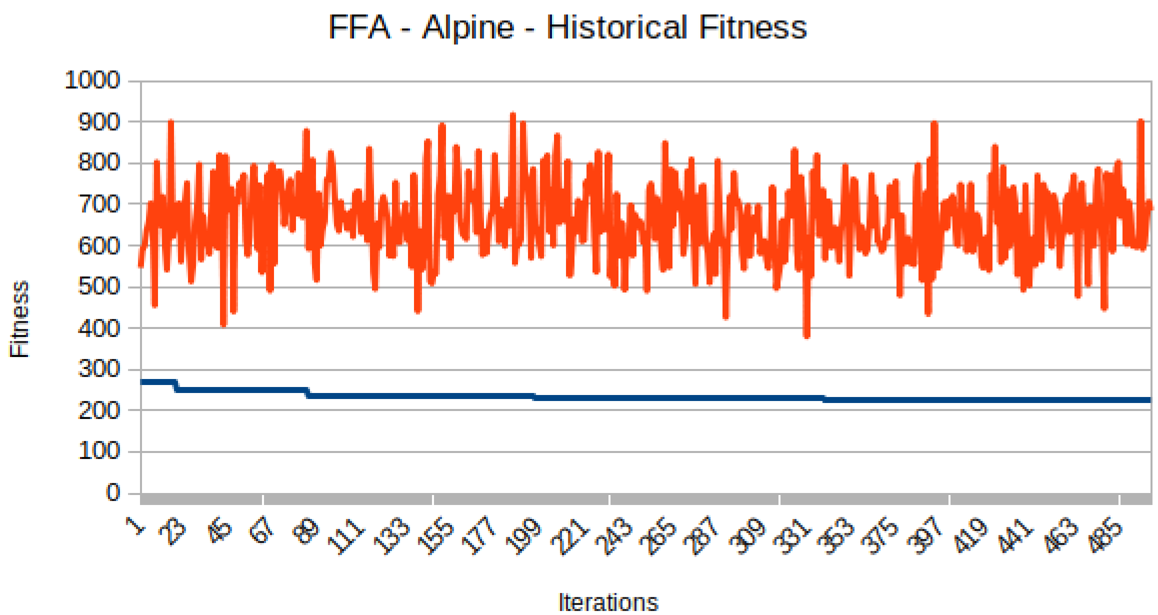


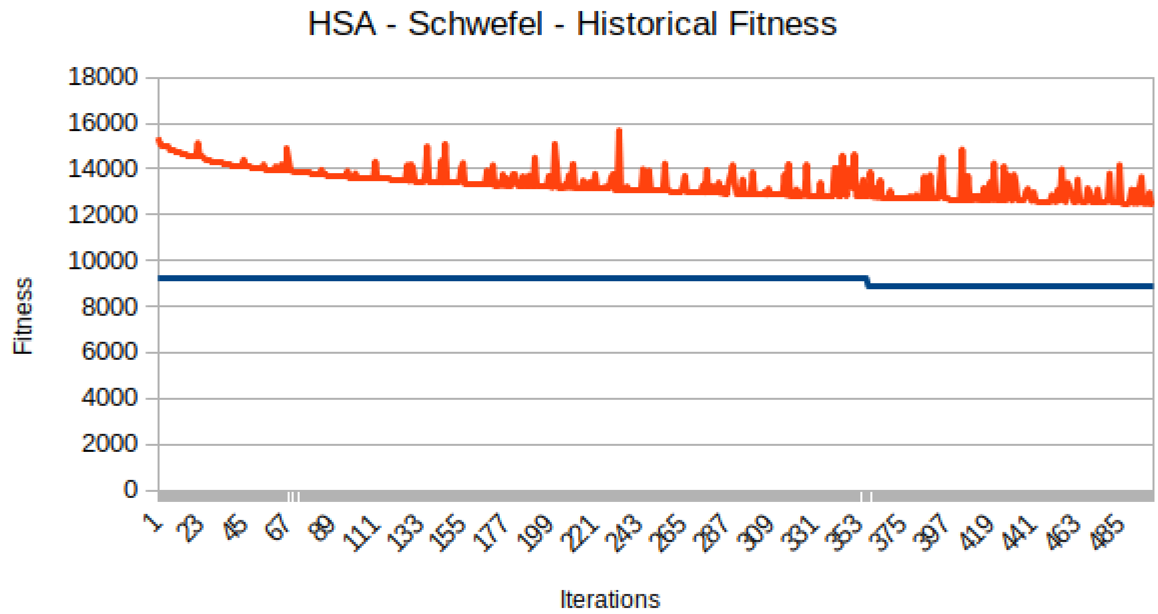
FFA - Quartic - Historical Fitness
Fitness
1800000000
1600000000
1400000000
1200000000
1000000000
800000000
600000000
400000000
200000000
0
1
24
47
70
93
116
139
162
185
208
231
254
277
300
323
346
369
392
415
438
461
484
Iterations

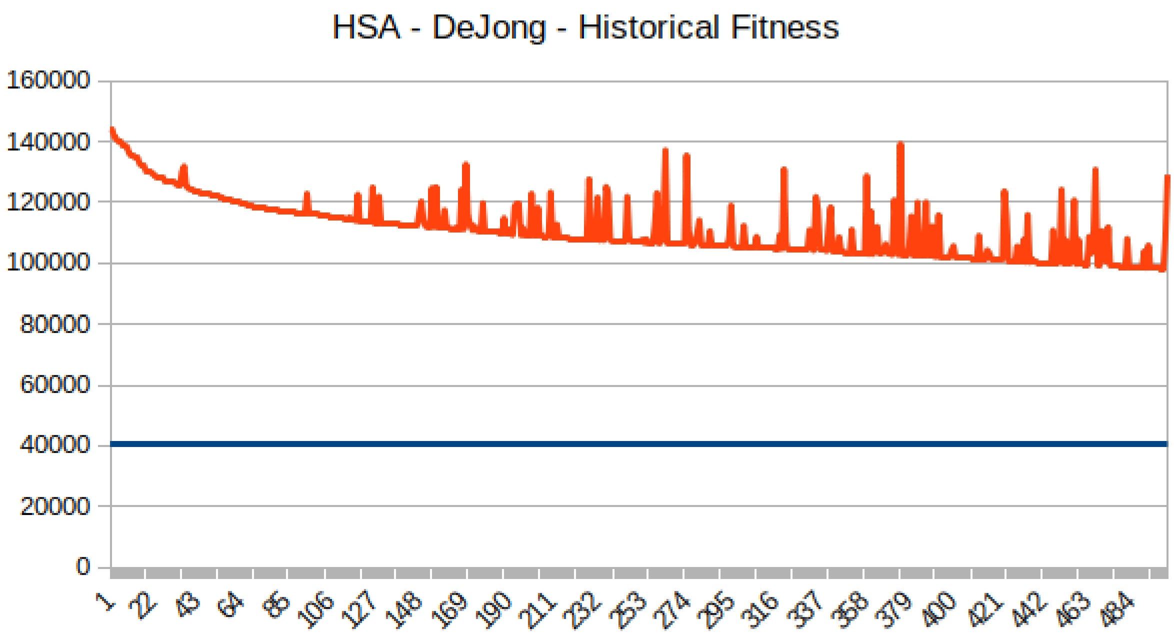


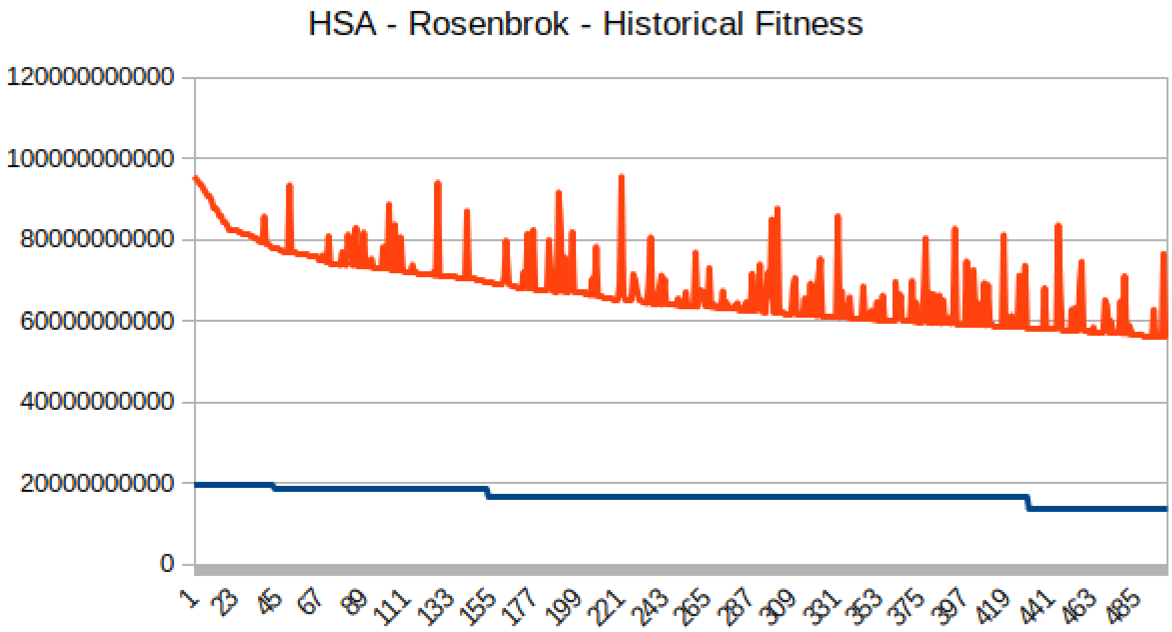
FFA - Levy - Historical Fitness
Fitness
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

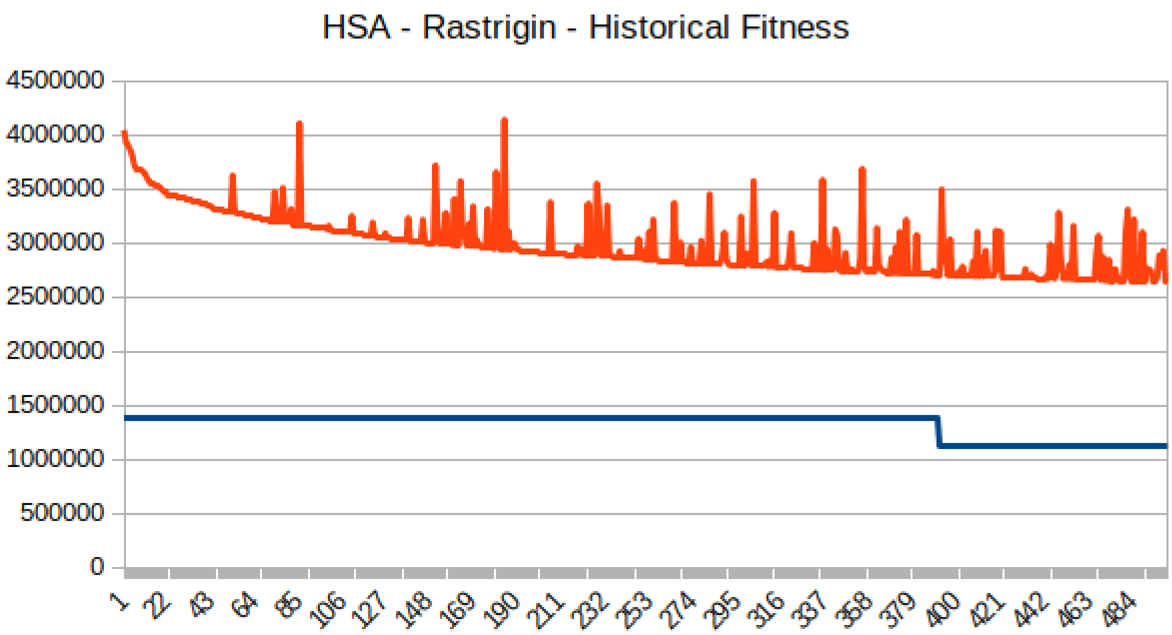

FFA - Step - Historical Fitness
Fitness
35000
30000
25000
20000
15000
10000
5000
0
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485
Iterations

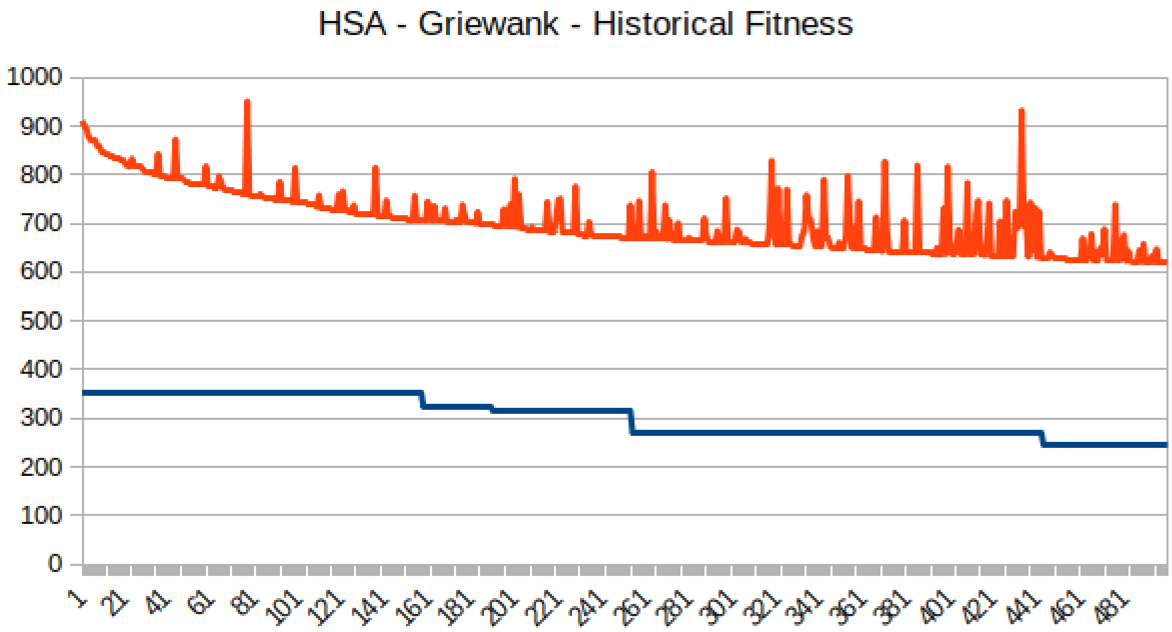

FFA - Alpine - Historical Fitness
Fitness
1000
900
800
700
600
500
400
300
200
100
0
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485
Iterations

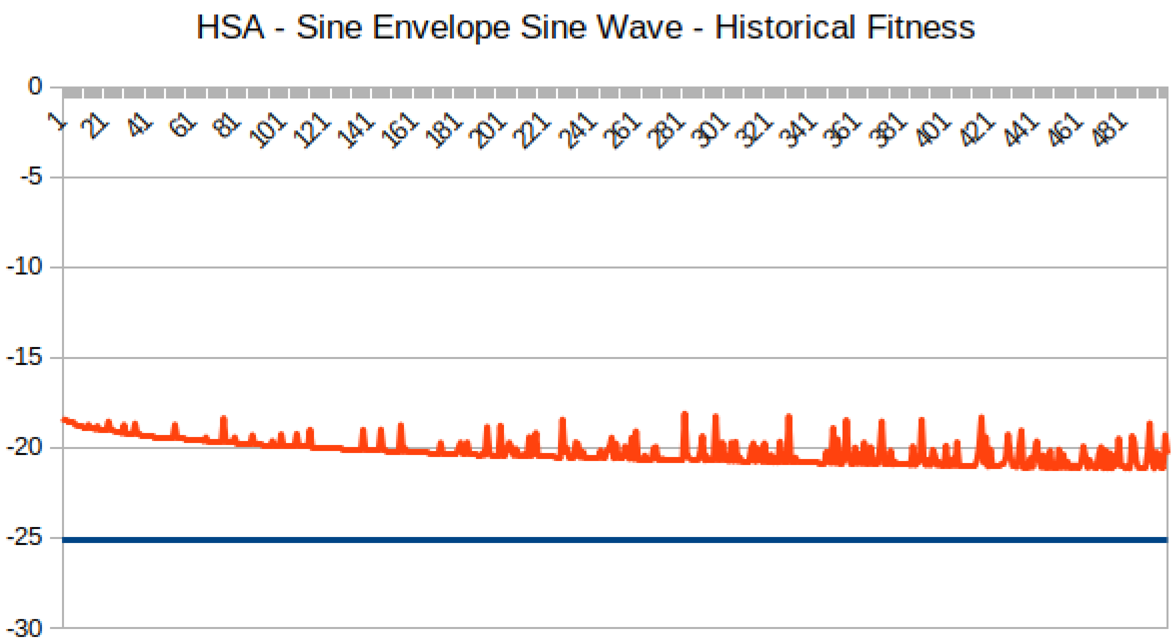

HSA - Schwefel - Historical Fitness
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Fitness
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485
Iterations

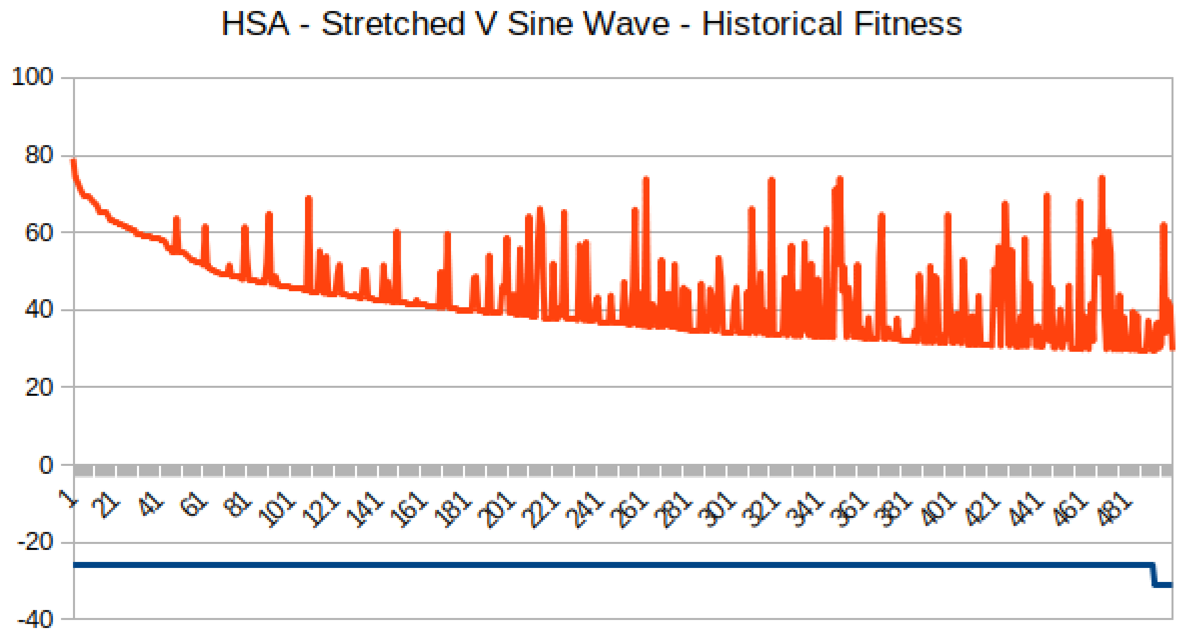

HSA - DeJong - Historical Fitness
160000
140000
120000
100000
80000
60000
40000
20000
0
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

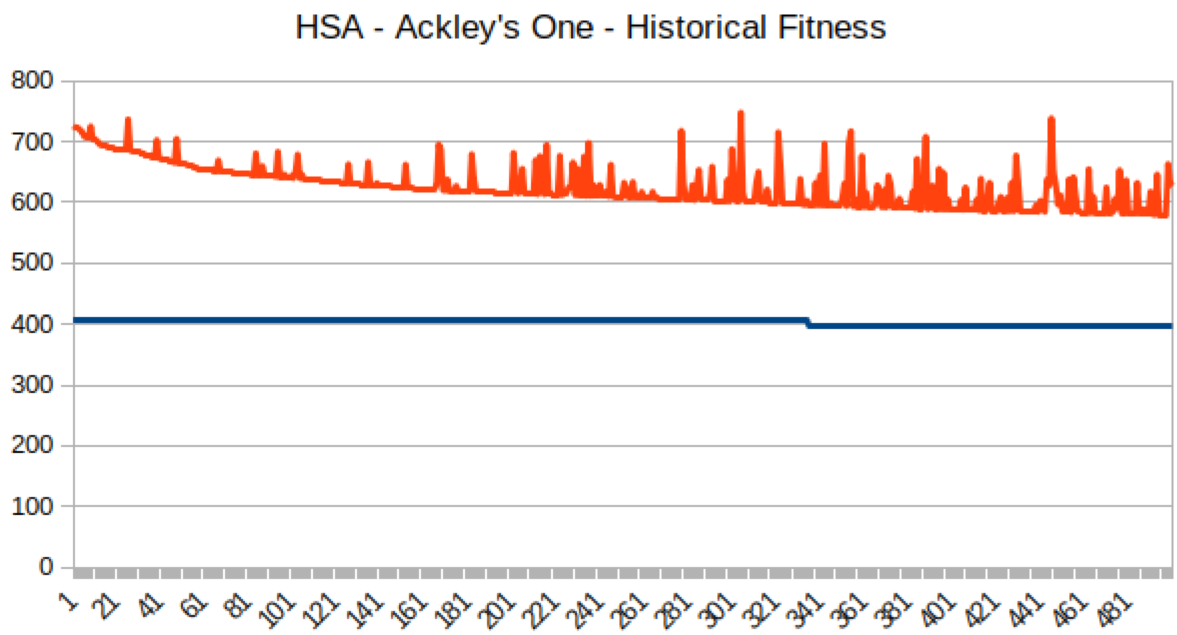

HSA - Rosenbrok - Historical Fitness
120000000000
100000000000
80000000000
60000000000
40000000000
20000000000
0
1
23
45
67
89
111
133
155
177
199
221
243
265
287
309
331
353
375
397
419
441
463
485

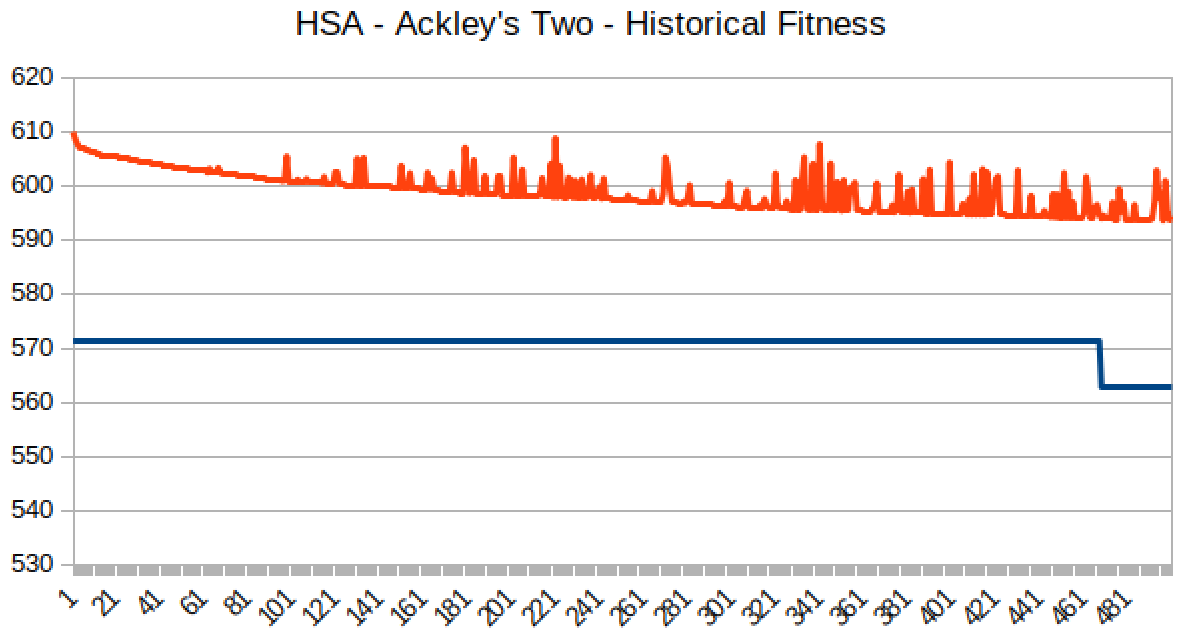

HSA - Rastrigin - Historical Fitness
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
1
22
43
64
85
106
127
148
169
190
211
232
253
274
295
316
337
358
379
400
421
442
463
484

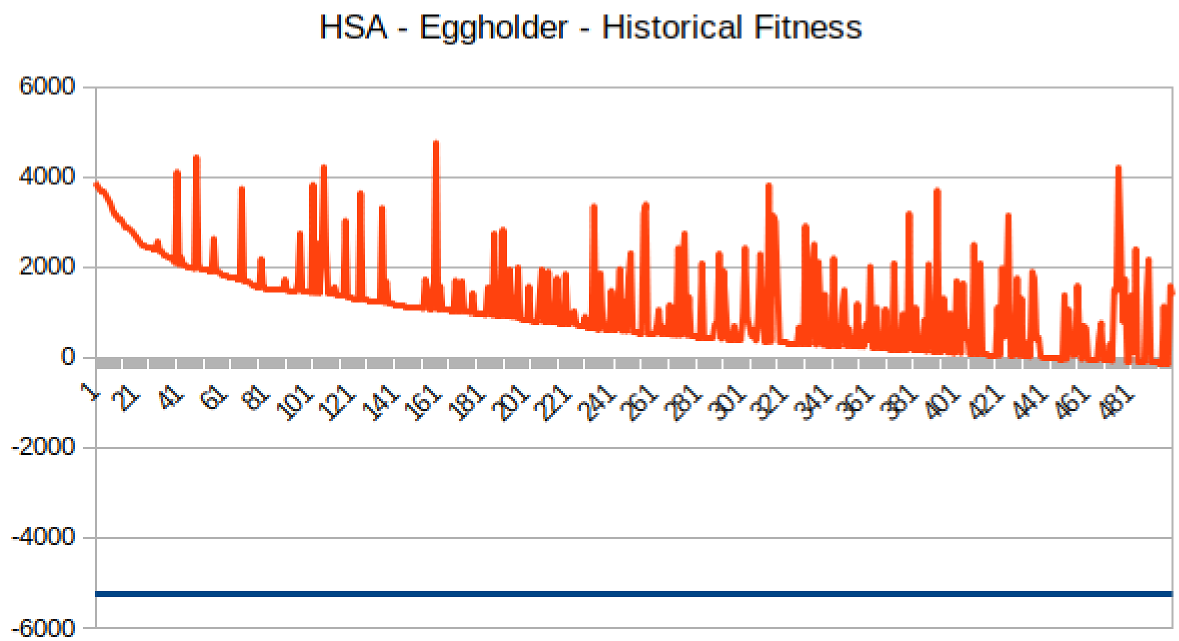

HSA - Griewank - Historical Fitness
1000
900
800
700
600
500
400
300
200
100
0
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481

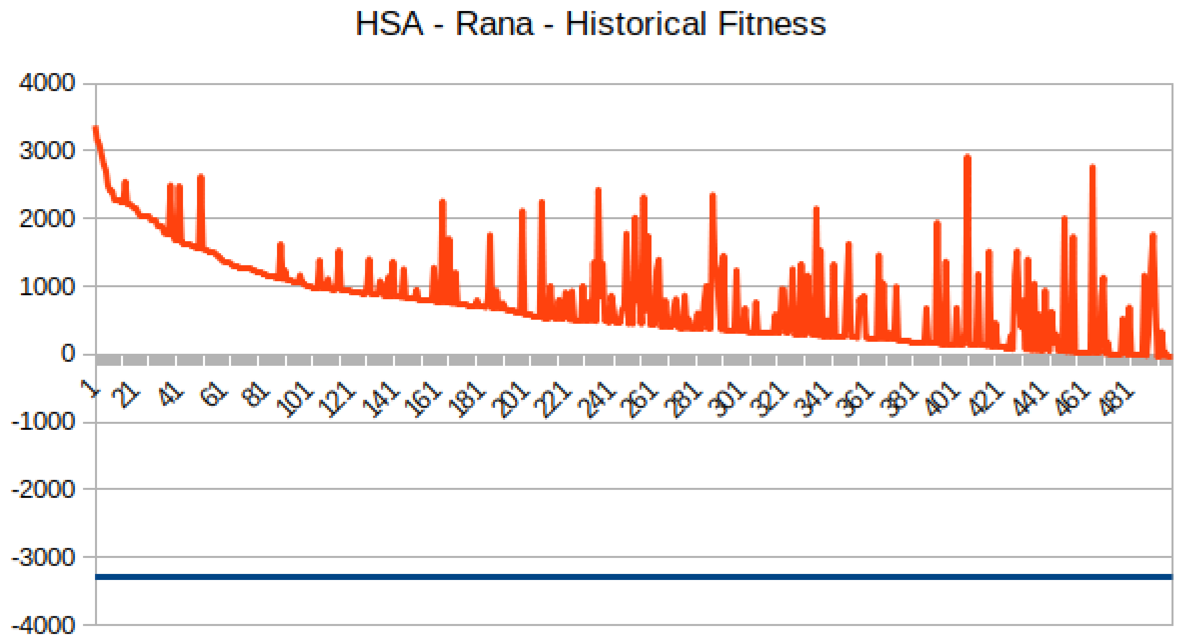

HSA - Sine Envelope Sine Wave - Historical Fitness
0
-5
-10
-15
-20
-25
-30
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481

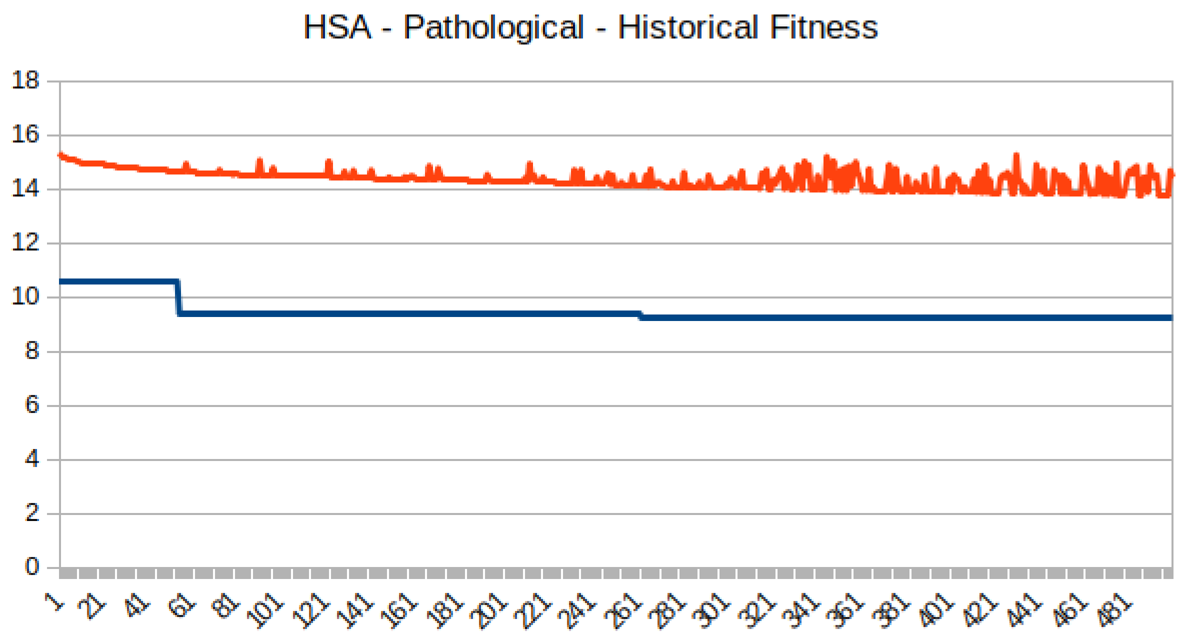

HSA - Stretched V Sine Wave - Historical Fitness
100
80
60
40
20
0
-20
-40
1 21 41 61 81 101 121 141 161 181 201 221 241 261 281 301 321 341 361 381 401 421 441 461 481


HSA - Ackley's One - Historical Fitness
800
700
600
500
400
300
200
100
0
1 21 41 61 81 101 121 141 161 181 201 221 241 261 281 301 321 341 361 381 401 421 441 461 481

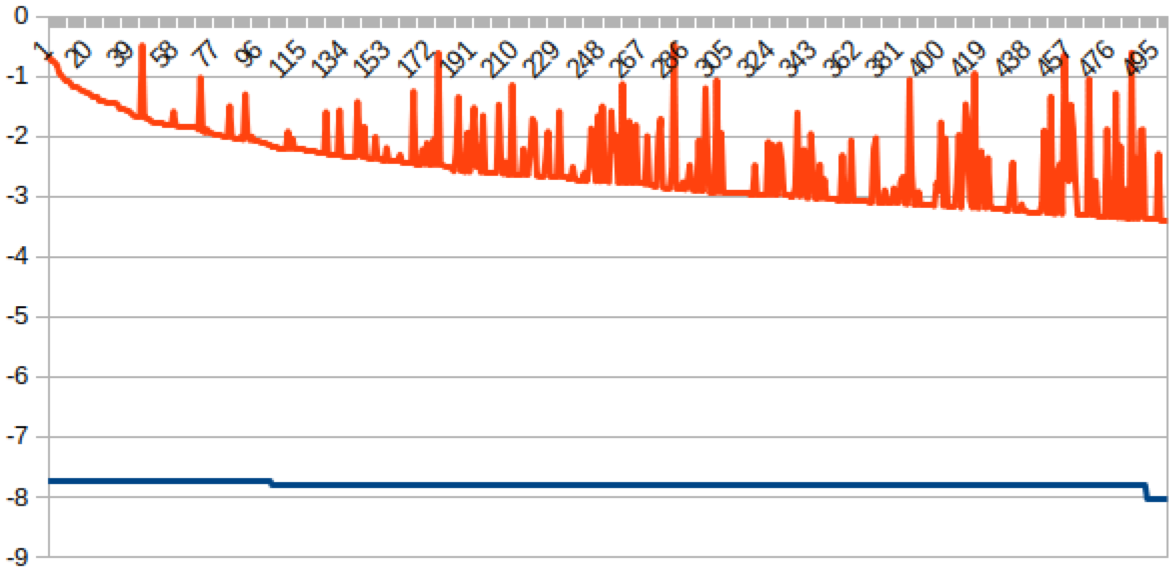

HSA - Ackley's Two - Historical Fitness
620
610
600
590
580
570
560
550
540
530
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481


HSA - Eggholder - Historical Fitness
6000
4000
2000
0
-2000
-4000
-6000
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481

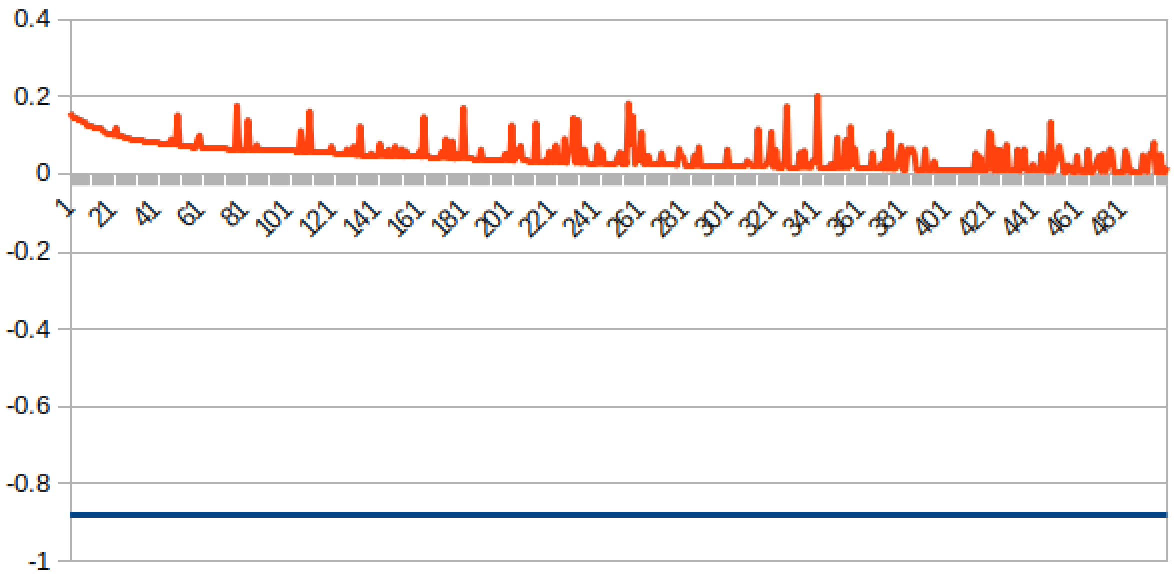

HSA - Rana - Historical Fitness
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481

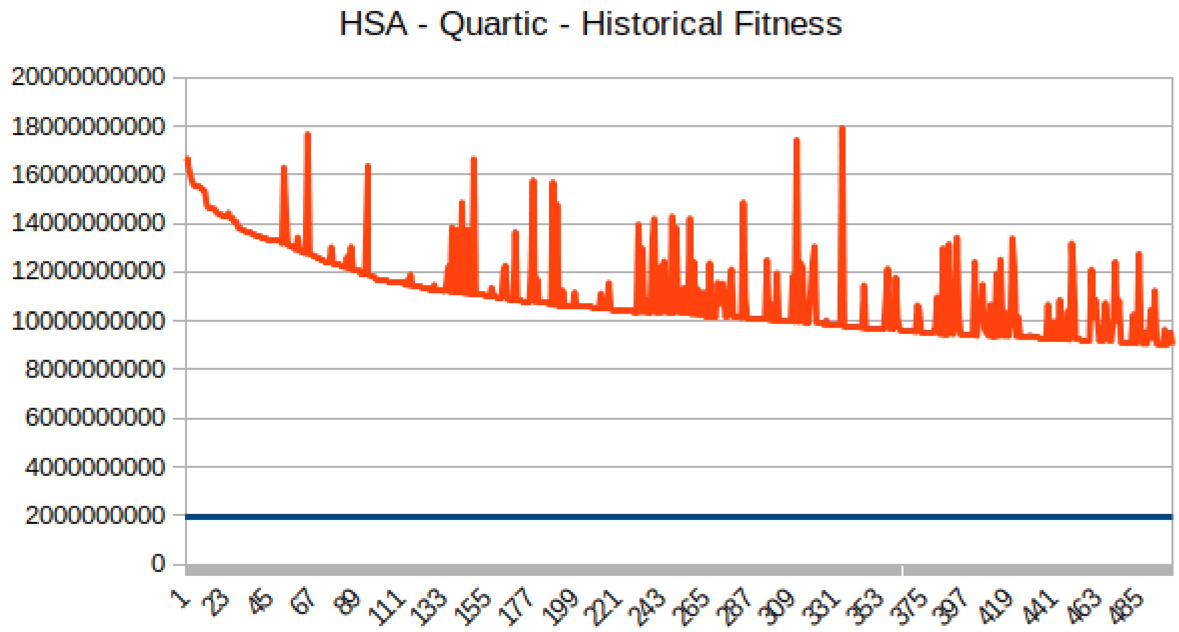

HSA - Pathological - Historical Fitness
18
16
14
12
10
8
6
4
2
0
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481

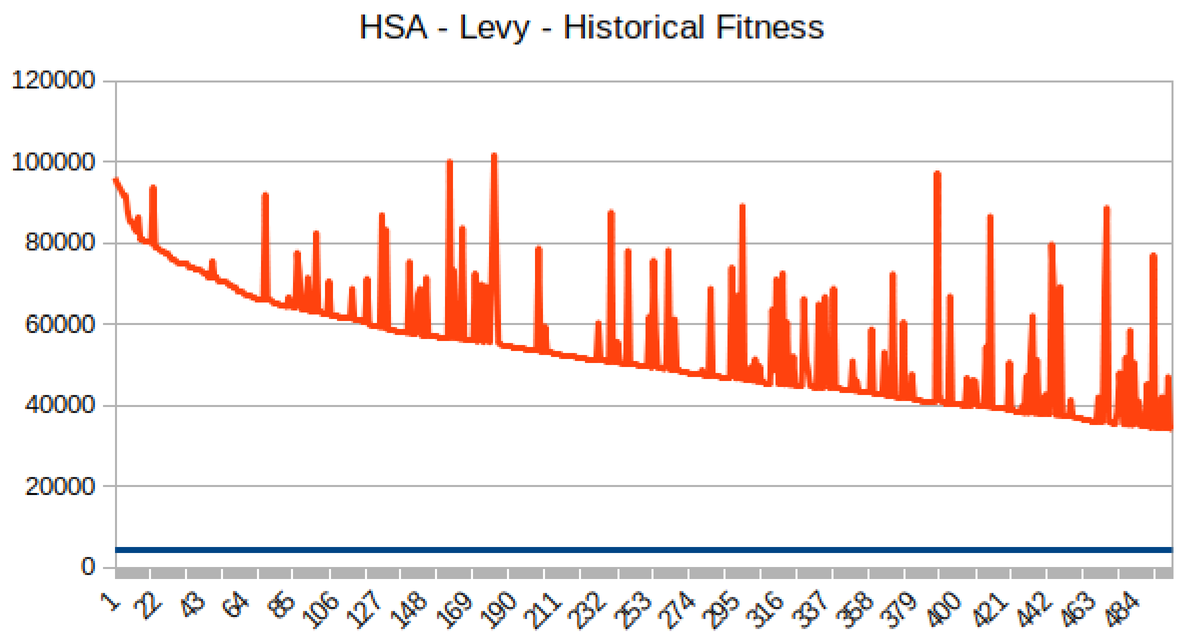
HSA - Michalewicz - Historical Fitness



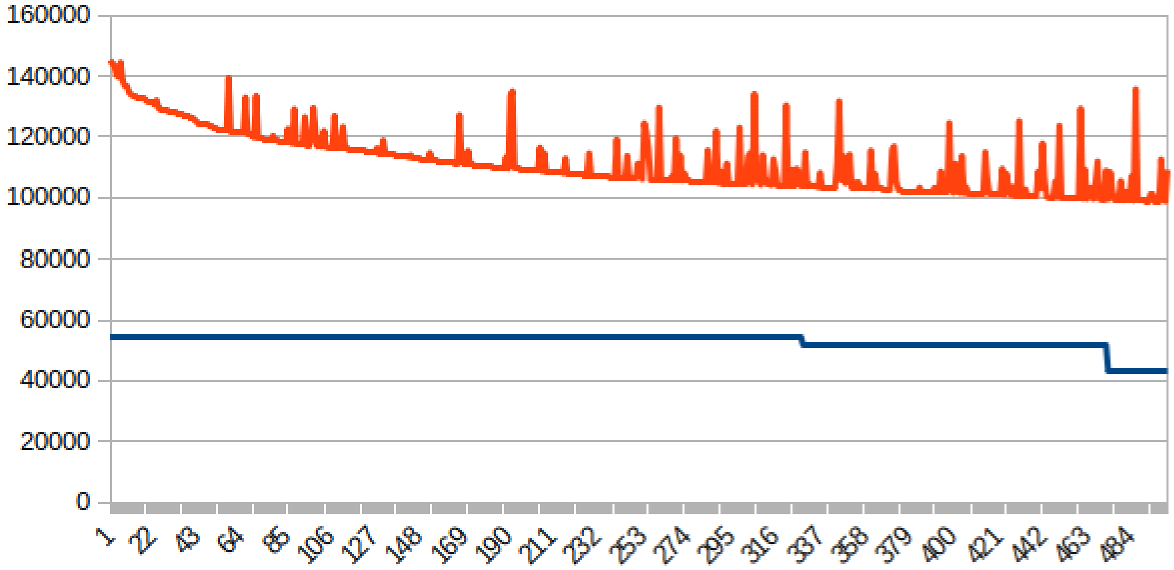
HSA - Master's Cosine Wave - Historical Fitness




HSA - Quartic - Historical Fitness
20000000000
18000000000
16000000000
14000000000
12000000000
10000000000
8000000000
6000000000
4000000000
2000000000
0
1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485


HSA - Levy - Historical Fitness
120000
100000
80000
60000
40000
20000
0
1 22 43 64 85 106 127 148 169 190 211 232 253 274 295 316 337 358 379 400 421 442 463 484

HSA - Step - Historical Fitness



HSA - Alpine - Historical Fitness

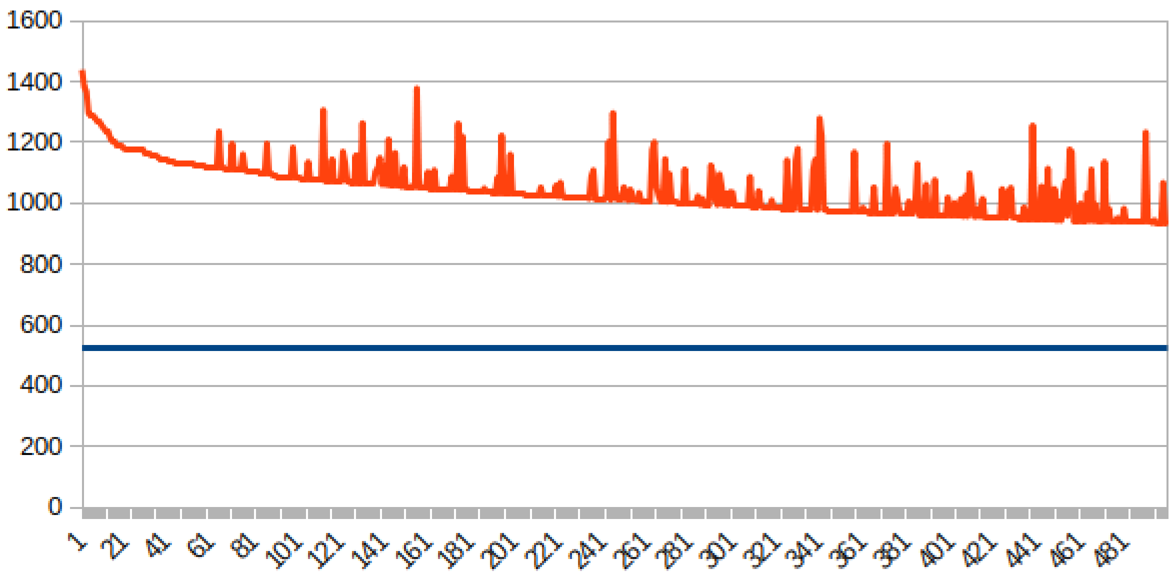


Table 1: Particle Swarm Final Fitness Statistics

Function	Average	Min	Median	Max	STDev
Schwefel	12503.31172	8886.46	12670.2	15007.4	1230.64359099699
DeJong	21.4825000000001	21.4825	21.4825	21.4825	0
Rosenbrok	6818.91999999996	6818.92	6818.92	6818.92	4.55202781497238E-11
Rastrigin	7667.14571840001	-42662.9	14334.55	46097.7	23347.9438460225
Griewank	1.52502000000001	1.52502	1.52502	1.52502	9.55748027557678E-15
Sin Envelope	-27.437192	-35.3501	-27.5517	-22.0314	1.74265978953441
Stretched V Sine	31.24056722	-44.4639	31.2199	77.3658	15.0250597242217
Ackley's 1	5.79036680000003	-24.4039	-24.4039	122.832	45.5357706562089
Ackley's 2	396.004019999998	392.982	392.982	471.735	11.8478316122159
Eggholder	323.88542268	-6403.81	535.4005	3863.87	1809.66683149914
Rana	275.450506418	-4476.89	365.1875	3655.25	999.874095676781
Pathological	8.85319241999999	-1.527	8.67917	15.975	2.6566964082874
Michalewicz	-6.22113161800001	-13.7017	-5.808305	-0.930869	2.32224295351277
Master's Cosine	-2.4351361978	-8.26384	-0.03594485	0.799412	3.81709477737896
Quartic	1154.64000000001	1154.64	1154.64	1154.64	1.06972653651851E-11
Levy	14428.49564	1409.64	15609.25	25930.6	5061.68429538422
Step	87.4319000000006	87.4319	87.4319	87.4319	5.54778389949759E-13
Alpine	82.0946426	32.8723	93.35055	167.508	33.6162214124096

Table 2: Firefly Final Fitness Statistics

Function	Average	Min	Median	Max	STDev
Schwefel	4304.61044	3279.51	4337.32	12516.6	419.900785135395
DeJong	1494.888826	950.627	1514.385	5742.63	220.032841131101
Rosenbrok	17400123.26	9083190	17494550	248839000	10620828.7793882
Rastrigin	33113.1348	10568.3	34216.1	176922	7943.2137337774
Griewank	9.84039037999999	5.63688	9.994205	69.0377	2.77327380080997
Sin Envelope	-30.3005964	-31.3991	-30.2395	-24.8076	0.355112357430489
Stretched V Sine	-116.0978332	-128.303	-115.704	-5.7916	5.38342728517506
Ackley's 1	78.8552238	42.8897	79.9184	182.161	6.26792457680008
Ackley's 2	465.442646	449.617	466.177	537.538	4.76206524266886
Eggholder	-10051.31874	-12394.6	-9953.995	-3330.42	453.879732083075
Rana	-13453.3511200001	-13595.3	-13463.9	-4554.86	398.973434022941
Pathological	-1.95124148	-5.1876	-1.766755	13.2387	0.983523159810269
Michalewicz	-11.33956334	-13.206	-11.257	-6.50987	0.393878948630404
Master's Cosine	-4.5126203786	-5.72495	-4.442825	0.0574107	0.318258797841235
Quartic	3376360.34	1644890	3348045	79979700	3459536.76692199
Levy	52.2047026	28.3501	52.0464	632.622	26.2402464038126
Step	1657.0773	1086.65	1678.395	7724.49	294.199126818132
Alpine	329.963072	260.904	333.548	658.707	21.2350723303823

Table 3: Harmony Search Final Fitness Statistics

Function	Average	Min	Median	Max	STDev
Schwefel	11602.86428	9373.93	11744.25	12410.6	636.530677715558
DeJong	85206.7758	51104.5	86941.4	98589.2	9601.3965201354
Rosenbrok	45468779600	19210100000	46590000000	64125100000	7657061067.5669
Rastrigin	2312085.12	1366370	2384360	2890590	261499.297808798
Griewank	537.153678	352.637	545.7055	787.986	57.8643309185167
Sin Envelope	-22.280004	-25.3194	-22.12	-21.1919	0.796263046123103
Stretched V Sine	14.4024578518	-27.69	16.60345	29.6141	11.4697317736656
Ackley's 1	529.498064	412.597	535.9825	580.51	36.5654932628978
Ackley's 2	588.10799	565.617	589.5455	605.457	5.22401442526058
Eggholder	-1279.6314238	-5514.99	-1031.78	506.864	961.573208955662
Rana	-772.4714763	-3129.13	-631.0575	67.9509	642.665394621495
Pathological	12.78424844	8.23381	13.03455	14.8979	0.847243167125027
Michalewicz	-4.43042552	-7.52787	-4.201955	-3.34857	0.869715623461436
Master's Cosine	-0.086495593342797	-1.74212	6.66509E-06	0.00422726	0.219766494345669
Quartic	7073521660	1928000000	7292560000	8994680000	1350931602.57265
Levy	19637.18672	3865.98	19849.75	33793.2	8240.61095663179
Step	85843.7842	41064.7	88336.55	98934.8	9981.85333427738
Alpine	840.993965999999	389.069	855.6375	1121.43	83.515689578956

Particle Swarm consistently did not improve any of the populations. This is most likely due to the constants being non-optimal for most of the functions. The few base functions that saw improvements only changed by a small amount. Additionally, the worst fitness did not see any improvements over time. The range of the worst fitness was random throughout the full runtime, but it's range of values was consistently within the same range. The exception was in the Rana function, where two-thirds of the way through the execution, the worst fitness rose by about 1000 on average.

The Firefly algorithm saw some good initial improvement in the fitness, but stagnated fairly early on. However, some functions did see some additional minor improvements later in the execution, suggesting that the algorithm would continue improve with more iterations. The worst fitness followed a similar trend to the best fitness, where the average value of the worst fitness also improved over time. This would provide not just a single best value in the population, but a collection of solutions that are approaching the optimal value.

Harmony Search did not see many improvements in the best fitness in the five hundred iterations, but there were improvements in most of the base functions. The constantly decreasing worst fitness indicates that better solutions are being continuously generated, with the spikes being solutions that were worse than the current worst. This, along with the occasional improvements in the best fitness, suggests that most of the new generated solutions are better than the worst, but not always better than the current best solution. Using a larger amount of iterations may show further, consistent improvement in the values of the populations.

The analysis of the final fitnesses of the population can provide more information about the algorithm itself. Particle Swarm shows many function with a very low standard deviation, with some being very close to 0. This suggests that there was a lot of stagnation that happened, where most of the results converged onto the same point to not be further improved.

Firefly showed many improvements in fitnesses over both Particle Swarm and Harmony Search across the board. The standard deviation was consistently low through each base function, and the resulting fitnesses were generally more optimal than those found in the other algorithms.

Harmony Search featured relatively low standard deviations for each base function, with the deviations being higher than Firefly but lower than Particle Swarm. The resulting fitnesses were also generally higher than those found in the other two algorithms. This indicates that all of the solution in the population are being improved, but the best does not necessarily improve very often. Once again, testing with a larger amount of iterations may show further improvements in the fitnesses of the entire population.

3 Function Calls

One of the factors that determines whether an algorithm should be considered for use is how many function calls the algorithm performs. For intensive base functions, they should ideally be called few times, as more function calls means a longer run time. However, if a slower execution speed is acceptable, then a slower, more effective algorithm would be ideal. The following tables list some statistics about the three optimization algorithms' function calls per iteration.

Table 4: Particle Swarm Algorithm's Function Calls per Iteration

Function	Average	Min	Median	Max	STDev
Schwefel	25000	25000	25000	25000	0
DeJong	25000	25000	25000	25000	0
Rosenbrok	25000	25000	25000	25000	0
Rastrigin	25000	25000	25000	25000	0
Griewank	25000	25000	25000	25000	0
Sin Envelope	25000	25000	25000	25000	0
Stretched V Sine	25000	25000	25000	25000	0
Ackley's 1	25000	25000	25000	25000	0
Ackley's 2	25000	25000	25000	25000	0
Eggholder	25000	25000	25000	25000	0
Rana	25000	25000	25000	25000	0
Pathological	25000	25000	25000	25000	0
Michalewicz	25000	25000	25000	25000	0
Master's Cosine	25000	25000	25000	25000	0
Quartic	25000	25000	25000	25000	0
Levy	25000	25000	25000	25000	0
Step	25000	25000	25000	25000	0
Alpine	25000	25000	25000	25000	0

Table 5: Firefly Algorithm's Function Calls per Iteration

Function	Average	Min	Median	Max	STDev
Schwefel	124488.076	679	124750	124750	5549.238
DeJong	124750	124750	124750	124750	0
Rosenbrok	124750	124750	124750	124750	0
Rastrigin	124750	124750	124750	124750	0
Griewank	124024.284	615	124750	124750	9356.285
Sin Envelope	124750	124750	124750	124750	0
Stretched V Sine	124750	124750	124750	124750	0
Ackley's 1	124750	124750	124750	124750	0
Ackley's 2	124750	124750	124750	124750	0
Eggholder	124107.008	505	124750	124750	8324.179
Rana	103588.102	1566	106889.5	124750	19733.72
Pathological	124750	124750	124750	124750	0
Michalewicz	124750	124750	124750	124750	0
Master's Cosine	124750	124750	124750	124750	0
Quartic	124750	124750	124750	124750	0
Levy	124750	124750	124750	124750	0
Step	124750	124750	124750	124750	0
Alpine	124750	124750	124750	124750	0

Table 6: Harmony Search Algorithm's Function Calls per Iteration

Function	Average	Min	Median	Max	STDev
Schwefel	1	1	1	1	0
DeJong	1	1	1	1	0
Rosenbrok	1	1	1	1	0
Rastrigin	1	1	1	1	0
Griewank	1	1	1	1	0
Sin Envelope	1	1	1	1	0
Stretched V Sine	1	1	1	1	0
Ackley's 1	1	1	1	1	0
Ackley's 2	1	1	1	1	0
Eggholder	1	1	1	1	0
Rana	1	1	1	1	0
Pathological	1	1	1	1	0
Michalewicz	1	1	1	1	0
Master's Cosine	1	1	1	1	0
Quartic	1	1	1	1	0
Levy	1	1	1	1	0
Step	1	1	1	1	0
Alpine	1	1	1	1	0

It should be noted that not featured in these charts are the function calls performed when each population has initialized. This initialization process cost each algorithm the same amount of function calls. The amount of calls in this instance is the same as the size of the population, 500 in this test. These were not included in the graph due to the skew they would add to the resulting statistics. This skew would be most noticeable in algorithms with a low amount of function calls, where the statistics would reflect the iterations to be more expensive than they are in reality.

The most important aspect to note is the uniformity of values produced by a couple of the algorithms. For each iteration, the algorithms would undergo the same amount of function calls each time, with no deviation. This is a useful property when a function needs to have consistent performance throughout the course of its execution.

Particle Swarm produced that largest amount of function calls throughout its execution. It consistently ran 250,000 function calls for every iteration of the algorithm. With a population containing 500 particles, it will check each particle against every particle, for 500^2 function calls per iteration. There were 500 iterations ran during this test, which would bring the total to 125,000,000, and including the function calls during initialization, 125,000,500. This is a massive amount of function calls to perform, and this is evident when the base function calls are already inherently expensive.

Firefly, while not being as expensive as Particle Swarm, was also a fairly expensive algorithm. Each iteration performed an average of 61,741,874 function calls. This is roughly half of the function calls made by Particle Swarm, as Firefly will only create a new firefly about half of the time. There was some deviation present in Firefly that was not present in the other algorithms. For example, with the Schwefel function, while the maximum amount of function calls was 124,750, the lowest amount was 679. However, with the median being the same as the maximum value, the average being very close to those, and the standard deviation being relatively low in comparison, it is safe to say that one should assume that each iteration will perform the 124,750 function calls.

On the other hand, Harmony Search performs exactly one function call per iteration. This is because the production and acceptance of new solution vectors is not caused by comparing one harmony against every other harmony, but because only one harmony is changed per iteration. One change means one function call, meaning there are 500^1 function calls, for a total of $500^1 + 500 = 1000$. This is an amount that is magnitudes smaller than both Particle Swarm and Firefly, which is very useful when a low amount of function calls is required.

4 Elapsed Time per Iteration

The time an algorithm takes to fully execute is an important metric that needs to be taken into account. When an algorithm takes substantially longer than another one to produce similar results, then it should not be considered for future implementation. The following tables list the statistics for the run times of each function being ran through each optimization algorithm.

Table 7: Particle Swarm Algorithm's Time Elapsed per Iteration

Function	Average	Min	Median	Max	STDev	Total Runtime
Schwefel	14525.46292	8079.63	15546	19535.7	3321.568	7262731.46
DeJong	12155.61442	7450.85	13007	15239.9	1966.309	6077807.21
Rosenbrok	8024.46536	5581.41	7886.77	9240.78	502.0705	4012232.68
Rastrigin	9566.96746	7562.22	9507.83	11393.1	681.8025	4783483.73
Griewank	9175.92286	7653.11	9079.97	11223.1	588.1880	4587961.43
Sin Envelope	14863.30912	7369.41	16188.75	20616.4	3789.571	7431654.56
Stretched V Sine	16940.97742	1529.18	16346.25	33465.8	10612.301	8470488.71
Ackley's 1	11355.60266	6534.53	11698.95	13980.2	1478.566	5677801.33
Ackley's 2	16745.72668	2807.88	17640.5	29506.1	8319.463	8372863.34
Eggholder	14579.44208	7676.65	15715.15	19867	3452.208	7289721.04
Rana	16477.4676	5183.14	16901.9	26479.4	6648.934	8238733.8
Pathological	14966.94942	5835.52	15864.9	21235.5	3817.295	7483474.71
Michalewicz	15520.04938	5308.17	15291.15	22228	4559.525	7760024.69
Master's Cosine	16044.73206	5282.84	16273.2	24702	6033.651	8022366.03
Quartic	16602.28998	4103.88	16808.7	28809.2	7288.971	8301144.99
Levy	10068.54448	7192.07	10121.05	11916.6	943.2690	5034272.24
Step	7823.33842	6251.03	7665.9	8925.7	441.3720	3911669.21
Alpine	12272.80096	7540.39	13056.95	15274.8	1902.721	6136400.48

Table 8: Firefly Algorithm's Time Elapsed per Iteration

Function	Average	Min	Median	Max	STDev	Total Runtime
Schwefel	22354.197836	276.406	22760.9	24999	2292.58587389436	11177098.918
DeJong	20770.8014	18323.2	20778.15	23583.5	869.28441991298	10385400.7
Rosenbrok	20732.5312	16440.5	20756.8	22803.3	925.812418113749	10366265.6
Rastrigin	22284.359	16276.7	22550.6	24633.2	1574.00709994598	11142179.5
Griewank	22755.1107	1230.71	23355.65	26847.1	2645.95067439163	11377555.35
Sin Envelope	24290.25236	7180.54	26483.7	29701	5642.10005727433	12145126.18
Stretched V Sine	24969.39736	3583.8	30037.5	33878.7	9522.462528501	12484698.68
Ackley's 1	23038.0836	12158.2	23639.8	26975.2	2555.31039571476	11519041.8
Ackley's 2	24981.44836	1865.74	30229.95	35269.6	9704.33160841582	12490724.18
Eggholder	23629.69458	2155.04	25020.4	27974.5	4087.99097927578	11814847.29
Rana	24566.0538	1908.58	27693.5	31126.8	6841.98277736756	12283026.9
Pathological	23026.934	12343.5	23743.15	26670.5	2595.61125521917	11513467
Michalewicz	24743.35088	4928.06	28176.6	32726.7	7489.20082440963	12371675.44
Master's Cosine	24643.743	6700.52	27608.4	31420.1	6806.96102263301	12321871.5
Quartic	23938.20822	9232.57	25567.05	28917.4	4567.48441056538	11969104.11
Levy	21108.5028	16311.4	21173.9	23960.1	1052.88464073018	10554251.4
Step	20697.5278	16736	20730.5	22966.2	818.372561285684	10348763.9
Alpine	22240.1022	16637.2	22472.4	24924.8	1512.71558535619	11120051.1

Table 9: Harmony Search Algorithm's Time Elapsed per Iteration

Function	Average	Min	Median	Max	STDev	Total Runtime
Schwefel	1.046944	0.019	0.2415	76.429	6.5447	523.472
DeJong	0.886462	0.033	0.241	67.737	5.4940	443.231
Rosenbrok	0.577508	0.021	0.219	60.589	3.7988	288.754
Rastrigin	0.900828	0.022	0.225	88.006	6.4265	450.414
Griewank	0.95962	0.037	0.2275	90.691	6.9096	479.81
Sin Envelope	0.98453	0.038	0.229	71.997	4.9916	492.265
Stretched V Sine	0.710966	0.036	0.254	65.224	4.2647	355.483
Ackley's 1	1.01208	0.028	0.2375	95.985	6.6541	506.04
Ackley's 2	0.737682	0.031	0.254	57.841	4.5911	368.841
Eggholder	0.93852	0.028	0.251	91.798	6.2809	469.26
Rana	0.899428	0.026	0.236	90.968	6.0915	449.7140
Pathological	1.030984	0.028	0.247	57.457	5.6302	515.492
Michalewicz	0.881014	0.036	0.25	71.02	5.8575	440.507
Master's Cosine	0.881808	0.03	0.248	88.986	5.7631	440.904
Quartic	0.986568	0.038	0.2405	90.261	7.1588	493.284
Levy	1.058828	0.036	0.2295	103.575	7.2299	529.414
Step	1.002364	0.024	0.233	98.892	6.9558	501.182
Alpine	0.89165	0.03	0.24	89.748	6.2127	445.825

The grand total running time for each Algorithm is as follows: Particle Swarm ran for a cumulative 118,854,831.64 milliseconds, or approximately 33 hours and 54.8 seconds; Firefly ran for a cumulative total of 207,385,149.548 milliseconds, or approximately 57 hours, 36 minutes, and 25.1 seconds; Harmony Search ran for a cumulative total of 8,193.892 milliseconds, or approximately 8.2 seconds.

The reason both the Particle Swarm and Firefly algorithms have such long run times is because of their $O(n*m^2)$ base runtime complexity, as opposed to Harmony Search's O(n) complexity. As the number of solution vectors increases, the runtime for Particle Swarm and Firefly will exponentially increase, whereas the increase for Harmony Search is only linear.

Throughout each algorithm, the base functions do not seem to influence the overall runtime by a significant amount. Previous experiments showed that certain functions, such as Griewank, would run significantly longer than the the other functions. However, in this test there are no functions that stand out as being worse than the rest. This indicates that the bulk of the time spent in these algorithms is not influenced by the the base function calls, but by how many times the base function calls are made. This is further backed up by the standard deviation being fairly consistent between all the functions. There are some outliers, such as Step having a low standard deviation, or Ackley's One, with a very high deviation, but the total runtime shows that none of the functions run significantly longer (or shorter) than any of the others.

Furthermore, the averages and medians of all the functions are very similar, indicating that the data is fairly consistent. The max tends to not be much higher than either the average or the median as well, further backing up this claim. However, many of the minimum values are significantly lower than the aforementioned values. This is largely due to the iterations that did not produce many function calls. When an iteration does not make many function calls, the time will be much smaller, as the CPU will not be spending time running the base functions. However, with the average being close to the median, and the standard deviations being relatively small, it is safe to say that these small minimum values are outliers, and that there are not many of them.

For large populations, Harmony Search should be utilized if the runtime needs to be kept as small as possible. When it comes to smaller population sizes, then the differences in runtime won't be as noticeable. For small populations, any of the algorithms could be used confidently, but when the population grows larger, Harmony Search should be utilized, as the runtime will only grow linearly instead of exponentially.

5 Conclusion

There are many things to consider when deciding on an optimization algorithm to use. The key element to consider are: effectiveness in optimizing the population, time efficiency, and function call efficiency.

In this test, using the parameters stated earlier, the algorithm that produced the best results was the Firefly algorithm. While the algorithm may have stopped creating improvements fairly early on, it was more consistent and reliable than either Particle Swarm or Harmony Search. It should be noted than when increasing the number of iterations to large values such as 100,000, Harmony Search will continue to improve the solution throughout the iterations, but if the number of iterations being performed is limited to a smaller amount, its performance will suffer.

The clear best algorithm in terms of time efficiency is Harmony Search, were all of its functions executed in just a few seconds, magnitudes faster than either Particle Swarm or Firefly, which both had a cumulative total runtime of over a day. This speed is maintained when increasing both the population size, and the number of iterations the algorithm performs. If

Once again, in regards to function call efficiency, Harmony Search performed significantly better than either the Particle Swarm and Firefly algorithms. When function calls become very expensive to run, an algorithm that does not call them often would be preferable to use. Of these three algorithms, Harmony Search would be the ideal choice, as it made fewer calls through it's entire execution than almost every individual iteration of Firefly, and every single iteration of Particle Swarm.

On a final note, the effectiveness of these algorithms can be further improved by using separate values for the algorithmic constants for each function. While some functions will considerably improved using one set of constants, another function may not see any improvement at all. For future experiments, this should be considered to generate more optimized results.