CENTRAL WASHINGTON UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCE

Optimization using Particle Swarm, Firefly, and Harmony Search Algorithms

Emily Bodenhamer

May 20, 2019

CWU ID: 41119306

Dr. Donald Davendra

1 INTRODUCTION TO THE PROBLEM

The purpose for this project is to code three different meta-heuristics. Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Harmony Search Algorithm (HS) are the three meta-heuristics that were coded and had results extracted from. These algorithms are used to improve the fitness from eighteen different optimization functions. Thirty experimentations have been conducted on each algorithm with a minimum of 500 iterations. The average, standard deviation, median, range, time, and a calculation of the total number of calls to the fitness function are recorded for each algorithm.

1.1 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization is modeled after the flocking and schooling patters of birds and fishes. PSO was invented by Russell Eberhart and James Kennedy in 1995. A population must first be made up of random particles. The velocity and fitness of this population are calculated and a personal best and global best are found from the fitness of the population. The algorithm starts by updating the velocity based on constants that will move the particles towards either the personal best or global best. The velocity is used to update the population and the new updated population's fitness is calculated. If there is improvement for the personal best and/or global best, they are updated.

1.2 Firefly Algorithm

The Firefly Algorithm is modeled after the social behavior of fireflies based on the flashing and attraction characteristcs of fireflies. It was invented by Yang in 2010. For FA, the fitness function is associated with light intensity. If a firefly is brighter than the rest then it has the best fitness value. To search for a brighter firefly, attraction is calculated between different fireflies. The fireflies will move towards the brighter firefly. After a population of fireflies are created and the light intensity is calculated, the algorithm begins. The light intensity of fireflies are compared pair-wise. If one firefly is brighter than the other, the firefly moves towards the other, the attractiveness is calculated based on the distance from the two, and a new firefly is created and replaces the worst firefly in the population if its fitness value is better than the worsts.

1.3 Harmony Search

Harmony Search is based on the process of musicians when they improvise their instruments' pitch by searching for a perfect state of harmony. It was introduced in 2001 by Zong Woo Geem et al. For HS, a population of harmonics are created randomly. The fitness of these harmonics are calculated. New harmonies are created and added to the population based on two different strategies. New harmonies are created if a random number is less than a constant parameter called the harmony memory considering rate. If this criteria is met, then the harmonies are taken from a random harmony vector from the population. This value is then updated if another random number is less than another parameter, the pitch adjusting rate. If the random number was not less than the harmony memory considering rate, then new harmonies are added from obtaining a random number within the bounds of the objective function. If these new harmonies have a better fitness than the worst harmonics fitness in the population, then the new harmonies replace the worst harmonics.

1.4 Results and Analysis

Results from the three algorithms were recorded for each iteration and each experimentation. For each algorithm, 18 graphs have been made from each of the 18 functions. These graphs shows the evolution of the best fitness found throughout each iteration. Tuning the parameter values were used for different algorithms and functions.

1.5 Figure 1.1 Analysis

Figure 1.1 shows the evolution of the best solution found in the population from PSO for F(1). The best values to use to gain these results were k = 0.001, c1 = 0.3 and c2 = 0.7. The best solution initially starts at 9,000 and is continuously decreasing until iteration 229. In this iteration the population becomes stagnated and the minimum result gained were in the 3,000's.

1.6 Figure 1.2 Analysis

Figure 1.2 shows the evolution of the best solution found in the population from PSO for F(2). The best values to use to gain these results were k = 0.00001, c1 = 1.7 and c2 = 0.3. The best solution initially starts at 56200 and decreases until iteration 37. The best results were in the 52000's. Compared to the previous figure, the population had gotten stagnated more quickly resulting in the inability to get smaller results. In order to improve the results for this function even smaller k would need to be used to slower the velocity of the particles.

1.7 Figure 1.3 Analysis

Figure 1.3 shows the evolution of the best solution found in the population from PSO for F(3). The best values to use to gain these results were k = 0.00000000001, c1 = 0.3 and c2 = 0.7. The best solution initially starts at 1.8E+10 and decreases until iteration 151. The best results were in 1.3E+10. Compared to the previous figure, the population has also stagnated early on. The k value used for this function is very small, the need for this very small value concludes that the velocity of the particles are moving very fast causing stagnation to happen very early on.

1.8 Figure 1.4 Analysis

Figure 1.4 shows the evolution of the best solution found in the population from PSO for F(4). The best values to use to gain these results were k = 0.0000001, c1 = 0.2 and c2 = 0.6. The best solution initially starts at 1340000 and decreases until iteration 60. The best results were in 1290000. Compared to the previous figure, the population stagnated early on except not as early as figure 2. The k value used for this function is also very small. Functions that have large fitness values seem to require a very small k value to gain good results.

1.9 Figure 1.5 Analysis

Figure 1.5 shows the evolution of the best solution found in the population from PSO for F(5). The best values to use to gain these results were k = 0.001, c1 = 0.3 and c2 = 0.7. The best solution initially starts at 300 and continuously decreases. The best results recorded at iteration 500 is 160. This is the first function that has not come to a plateau. If more iterations were conducted for this function the best solution could still be improving.

1.10 Figure 1.6 Analysis

Figure 1.6 shows the evolution of the best solution found in the population from PSO for F(6). The best values to use to gain these results were k = 0.1, c1 = 0.2 and c2 = 0.6. The best solution initially starts at -20 and decreases until iteration 15. The best results recorded was -22.7. The value for k could be decreased to slower the velocity since the population stagnated quickly.

1.11 Figure 1.7 Analysis

Figure 1.7 shows the evolution of the best solution found in the population from PSO for F(7). The best values to use to gain these results were k = 0.2, c1 = 0.2 and c2 = 0.8. The best solution

initially starts at -22 and decreases until iteration 15. The best results recorded were -100. The plot for this figure is similar to figure 1.6, however the best solution decreased fasted in about the same number of iterations as the previous figure.

1.12 Figure 1.8 Analysis

Figure 1.8 shows the evolution of the best solution found in the population from PSO for F(8). The best values to use to gain these results were k = 0.0001, c1 = 0.2 and c2 = 0.8. The best solution initially starts at 380 and decreases until iteration 64. The best results recorded were 363. The value for k could be decreased to slower the velocity since the population stagnated quickly.

1.13 Figure 1.9 Analysis

Figure 1.9 shows the evolution of the best solution found in the population from PSO for F(9). The best values to use to gain these results were k = 0.001, c1 = 0.2 and c2 = 0.8. Stagnation in the population could have began at iteration 81 resulting in the fitness no longer improving

1.14 Figure 1.10 Analysis

Figure 1.10 shows the evolution of the best solution found in the population from PSO for F(10). The best values to use to gain these results were k = 0.0001, c1 = 0.2 and c2 = 0.8. After the first 20 iterations the best solution stays the same, however after 370 iterations a new best solution was found up until iteration 419 where the best solution flat lines again. More iterations for this function could provide better solutions if it repeats the same pattern seen in the past iterations.

1.15 Figure 1.11 Analysis

Figure 1.11 shows the evolution of the best solution found in the population from PSO for F(11). The best values to use to gain these results were k = 0.001, c1 = 0.2 and c2 = 0.8. The best fitness evolution of this plot starts off well, a smaller k could be used to have no stagnation in the population until a later iteration.

1.16 Figure 1.12 Analysis

Figure 1.12 shows the evolution of the best solution found in the population from PSO for F(12). The best values to use to gain these results were k = 1, c1 = 0.3 and c2 = 0.7.

1.17 Figure 1.13 Analysis

Figure 1.13 shows the evolution of the best solution found in the population from PSO for F(13). The best values to use to gain these results were k = 0.001, c1 = 0.3 and c2 = 0.7. The population for this graph stagnated quickly.

1.18 Figure 1.14 Analysis

Figure 1.14 shows the evolution of the best solution found in the population from PSO for F(14). The best values to use to gain these results were k = 0.01, c1 = 0.3 and c2 = 0.7. Compared with the previous function that had a smaller k value, the value for this function could have also been decreased to show improved results over a longer iteration time.

1.19 Figure 1.15 Analysis

Figure 1.15 shows the evolution of the best solution found in the population from PSO for F(15). The best values to use to gain these results were k = 0.000000001, c1 = 0.3 and c2 = 0.7. Similarly to the previous plot that has very large fitness values, a very small k was needed to see an improvement in the best solution for this plot as well.

1.20 Figure 1.16 Analysis

Figure 1.16 shows the evolution of the best solution found in the population from PSO for F(16). The best values to use to gain these results were k = 0.0001, c1 = 0.3 and c2 = 0.7.

1.21 Figure 1.17 Analysis

Figure 1.17 shows the evolution of the best solution found in the population from PSO for F(17). The best values to use to gain these results were k = 0.001, c1 = 0.3 and c2 = 0.7.

1.22 Figure 1.18 Analysis

Figure 1.18 shows the evolution of the best solution found in the population from PSO for F(18). The best values to use to gain these results were k = 0.0001, c1 = 0.3 and c2 = 0.7.

1.23 Figure 1.19 Analysis

Figure 1.19 shows the evolution of the best solution found in the population from FA for F(1). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.001.

1.24 Figure 1.20 Analysis

Figure 1.20 shows the evolution of the best solution found in the population from FA for F(2). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.01.

1.25 Figure 1.21 Analysis

Figure 1.21 shows the evolution of the best solution found in the population from FA for F(3). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 75.

1.26 Figure 1.22 Analysis

Figure 1.22 shows the evolution of the best solution found in the population from FA for F(4). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.5.

1.27 Figure 1.23 Analysis

Figure 1.23 shows the evolution of the best solution found in the population from FA for F(5). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.1.

1.28 Figure 1.24 Analysis

Figure 1.24 shows the evolution of the best solution found in the population from FA for F(6). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.1.

1.29 Figure 1.25 Analysis

Figure 1.25 shows the evolution of the best solution found in the population from FA for F(7). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.1.

1.30 Figure 1.26 Analysis

Figure 1.26 shows the evolution of the best solution found in the population from FA for F(8). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.1.

1.31 Figure 1.27 Analysis

Figure 1.27 shows the evolution of the best solution found in the population from FA for F(9). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.1.

1.32 Figure 1.28 Analysis

Figure 1.28 shows the evolution of the best solution found in the population from FA for F(10). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.001.

1.33 Figure 1.29 Analysis

Figure 1.29 shows the evolution of the best solution found in the population from FA for F(11). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.0001.

1.34 Figure 1.30 Analysis

Figure 1.30 shows the evolution of the best solution found in the population from FA for F(12). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.0001.

1.35 Figure 1.31 Analysis

Figure 1.31 shows the evolution of the best solution found in the population from FA for F(13). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.0001.

1.36 Figure 1.32 Analysis

Figure 1.32 shows the evolution of the best solution found in the population from FA for F(14). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.001.

1.37 Figure 1.33 Analysis

Figure 1.33 shows the evolution of the best solution found in the population from FA for F(15). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.00001.

1.38 Figure 1.34 Analysis

Figure 1.34 shows the evolution of the best solution found in the population from FA for F(16). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.0001.

1.39 Figure 1.35 Analysis

Figure 1.35 shows the evolution of the best solution found in the population from FA for F(17). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.01.

1.40 Figure 1.36 Analysis

Figure 1.36 shows the evolution of the best solution found in the population from FA for F(18). The best values to use to gain these results were alpha = 0.5, betamin = 0.2 and gamma = 0.01.

1.41 Harmony Search Analysis

The parameters used for HS is HMCR = 0.8, PAR = 0.3, and BW = 0.2. For every function in HS the worst fitness is always improving. The best fitness, however, never changes. The best fitness slightly changes for Stretched V Sine Wave, Michalewics, and Masters Cosine Wave. best fitness obtained from this algorithm does not perform as well as the other two algorithms.

1.42 Conclusion

After reviewing the results obtained from all three algorithms. FA had the best performance in minimizing the fitness value for all the functions within a small number of iterations. The results obtained from FA were able to get to the local minimum very quickly, within the first 50 iterations, for the 10 functions that have a local minimum of 0. The second best algorithm

Table 1.1: Particle Swarm Optimization Analytics

f(x)	Average	Standard Deviation	Median	Range	Time (ms)	Function Call
Schwefel	4634.597194	1776.680852	3442.602522	6213.293621	1091	249500
DeJong 1	51578.96884	957.2013967	51309.07176	3985.79369	968	249500
Rosenbrocks Saddle	20786769657	2109333175	19702283538	5334615619	2419	249500
Rastrigin	1331686.918	1522.913667	1330504.61	3148.016	1451	249500
Griewangk	208.8146687	26.26797293	205.108708	90.475603	1772	249500
Sine Envelope Sine Wave	-22.91717742	0.294695522	-22.960404	3.026892	4799	249500
Stretch V Sine Wave	-22.53543921	0.181477727	-22.558463	3.085485	4826	249500
Ackley One	316.1918392	10.59070695	313.026707	57.386537	5480	249500
Ackley Two	478.907586	5.039585632	477.040548	18.644082	3012	249500
Egg Holder	-5219.357186	620.6480812	-5716.940012	1395.794881	4396	249500
Rana	-9210.539488	3617.378882	-10701.2924	8926.023049	3261	249500
Pathological	-24.18549	6.939245082	-27.997573	27.490333	5136	249500
Michalewicz	-9.006188134	0.265701026	-9.095885	1.767938	2317	249500
Masters Cosine Wave	-3.516495559	1.453906749	-3.218388	3.744591	4404	249500
Quartic	2712809381	62820052.67	2705143081	651825554	882	249500
Levy	694.869074	0.318843792	694.816117	2.706602	940	249500
Step	408.8517643	21.84376109	393.922336	122.058311	864	249500
Alpine	9450.753823	542.6975577	9306.506257	3152.132133	4380	249500

was PSO. None of the results obtained from PSO were as close to the local minima as FA, however a majority of the results outperformed the algorithm used for the previous project, Differential Genetic Algorithm (DE). When comparing the analytic table of PSO with DE, a majority of the results outperform it, although some only slightly such as Ackley One and Two which only had a decrease in PSO of 24.3355808 and 13.828545. The last algorithm, HS, performed the worst out of all the algorithms, also performing worse than DE on certain functions. When comparing the time needed for the three algorithms, PSO would be a good choice to use. FA takes a long time to run and if a very large data set is used and a large amount of tests are required it could take a few days to run by using the same method of coding that was used for this report. A lot of improvements could be used on this algorithm to lessen the time needed for running the method, such as using threads or using openMP.

Table 1.2: Firefly Algorithm Analytics

f(x)	Average	Standard Deviation	Median	Range	Time (ms)	Function Call
Schwefel	435.8406077	1626.196268	0.000382	8270.664831	132734	30185067
DeJong 1	1176.958026	8185.817016	0	80456.17378	79187	20550816
Rosenbrocks Saddle	807711888	6525236984	0	1.1E+11	605545	62250250
Rastrigin	-9280.559731	424685.2355	-87000	3710256.506	396298	62250250
Griewangk	17.6912997	94.13722184	0	887.110198	486990	61880529
Sine Envelope Sine Wave	-40.77734983	3.867411861	-41.902081	21.785603	1272838	62250250
Stretch V Sine Wave	-151.1710104	16.84422139	-154.268467	123.65962	1175313	62250250
Ackley One	-53.84735446	95.58551921	-81.707625	709.403716	280175	26507136
Ackley Two	31.2964798	107.5888358	0.00035	540.501727	127162	8652771
Egg Holder	-19735.90754	3221.059427	-20570.46148	17993.54373	441489	62250250
Rana	-18405.7041	2614.217896	-19152.07438	15365.5761	130152	11617838
Pathological	-26.89642369	4.812558431	-28	33.979554	1110539	62250250
Michalewicz	-10.90971558	0.933301617	-11.140014	5.341727	599465	62250250
Masters Cosine Wave	-26.15602531	5.758282618	-28	27.262008	959443	62248735
Quartic	109259574.4	922546794.9	0	8954756468	228305	62250919
Levy	58.49530974	184.0129205	7.25001	1151.743184	224436	62250777
Step	15.11193527	69.71646917	0.000012	690.445928	214117	62250250
Alpine	341.2706783	1874.613566	3.5	14202.39764	721249	46217134

Table 1.3: Harmony Search Best Fitness Analytics

f(x)	Average	Standard Deviation	Median	Range	Time (ms)	Function Call
Schwefel	53633.47548	3.27E-10	53633.47548	0	196	500
DeJong 1	8994.047993	1.82E-11	8994.047993	0	183	500
Rosenbrocks Saddle	18651107556	0	18651107556	0	178	500
Rastrigin	1305518.426	6.98E-10	1305518.426	0	188	500
Griewangk	296.361594	1.14E-13	296.361594	0	184	500
Sine Envelope Sine Wave	-19.539197	5.33E-14	-19.539197	0	196	500
Stretch V Sine Wave	-20.09816915	3.221134111	-18.480397	8.360266	203	500
Ackley One	391.036384	3.98E-12	391.036384	0	194	500
Ackley Two	497.282639	4.60E-12	497.282639	0	195	500
Egg Holder	-4774.914144	29.31858371	-4768.929513	149.615771	203	500
Rana	-3272.868527	1.41E-11	-3272.868527	0	196	500
Pathological	2.176916	1.33E-15	2.176916	0	201	500
Michalewicz	-6.917574	3.29E-14	-6.917574	0	193	500
Masters Cosine Wave	-1.262299	6.88E-15	-1.262299	0	190	500
Quartic	3589882174	0	3589882174	0	185	500
Levy	564.343092	3.07E-12	564.343092	0	184	500
Step	521.322773	1.82E-12	521.322773	0	188	500
Alpine	13645.91617	8.19E-11	13645.91617	0	182	500

Table 1.4: Harmony Search Worst Fitness Analytics

f(x)	Average	Standard Deviation	Median	Range	Time (ms)	Function Call
Schwefel	132566.2179	6015.531805	130239.964	31752.0421	196	500
DeJong 1	13668.71954	363.0706911	13485.115	1854.74587	183	500
Rosenbrocks Saddle	70250547840	7341761395	65926379185	37523885873	178	500
Rastrigin	3021342.07	222336.599	2952025.455	1047154.387	188	500
Griewangk	741.4754404	42.19506601	723.3396965	207.021435	184	500
Sine Envelope Sine Wave	-13.07874786	1.015704045	-13.346987	5.277949	196	500
Stretch V Sine Wave	38.66762735	10.59620652	35.6921685	50.722427	203	500
Ackley One	603.5085898	27.23051258	599.827379	135.155234	194	500
Ackley Two	576.7241372	5.998081674	576.442719	28.984911	195	500
Egg Holder	797.932552	866.0250935	563.8680515	4284.289275	203	500
Rana	522.0320944	611.3680688	355.0322395	3047.679942	196	500
Pathological	10.96394215	1.134479191	10.5863715	4.804589	201	500
Michalewicz	-2.241627598	0.507554582	-2.351603	2.462429	193	500
Masters Cosine Wave	0.065694846	0.159489277	0.0004205	0.870562	190	500
Quartic	10913631841	1259845177	10255034639	6361374608	185	500
Levy	1316.965196	80.78771863	1281.394703	419.859675	184	500
Step	1134.342687	62.31744459	1113.789473	337.087954	188	500
Alpine	46955.41399	2748.482601	45981.7994	14646.42511	182	500

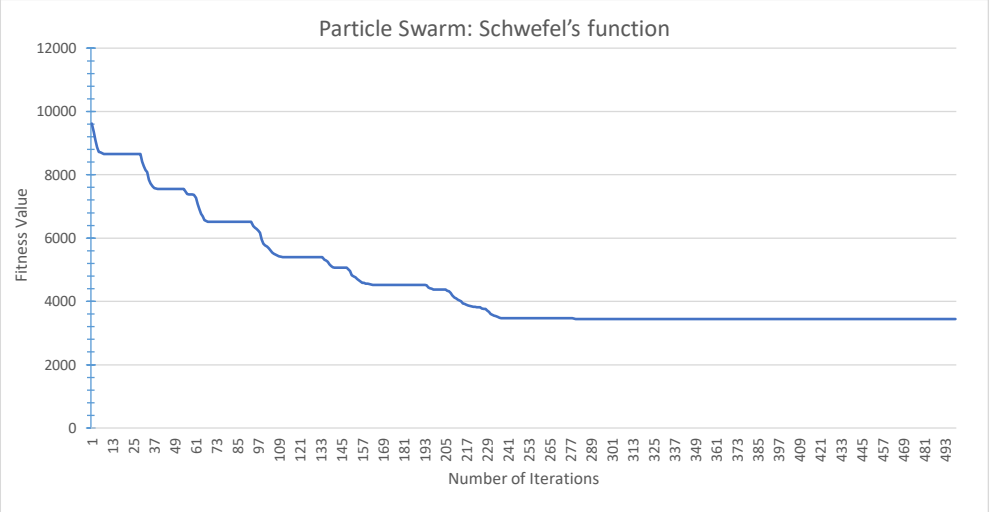

Figure 1.1: Best Fitness obtained using Particle Swarm Optimization on F(1)

Table 1.5: Statistical Analytics for Differential Algorithm for the Best Strategies

Average	StdDev	Time(msec)	DE strategy (1-10)
7920.003093	68.94682032	77	4
11849.20525	22.94100415	80	4
11845.59715	73.0235972	73	5
12287.11656	1.27E-11	74	5
288.633807	6.25E-13	78	1
-18.694878	1.42E-14	72	5
-8.563981	1.07E-14	78	1
340.52742	6.25E-13	72	2
492.736131	3.41E-13	80	2
-3858.456528	5.46E-12	84	2
-2304.712093	2.73E-12	91	9
3.186096	1.78E-15	77	4
-3.044226	0	96	8
-0.700846	9.99E-16	90	3
11853.03971	83.27710427	107	10
587.990224	5.68E-13	86	6
505.100898	8.53E-13	74	4
11868.15807	80.08025673	88	9

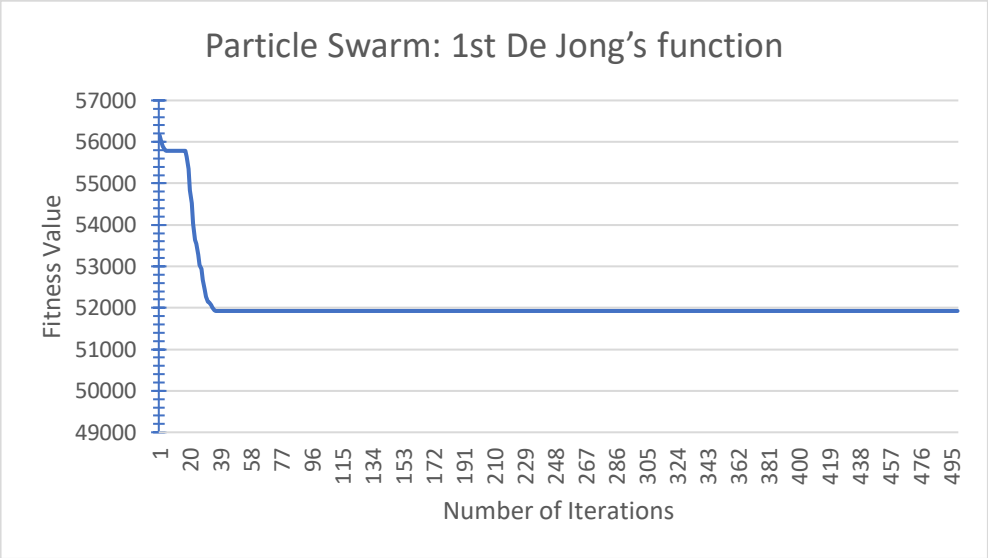

Figure 1.2: Best Fitness obtained using Particle Swarm Optimization on F(2)

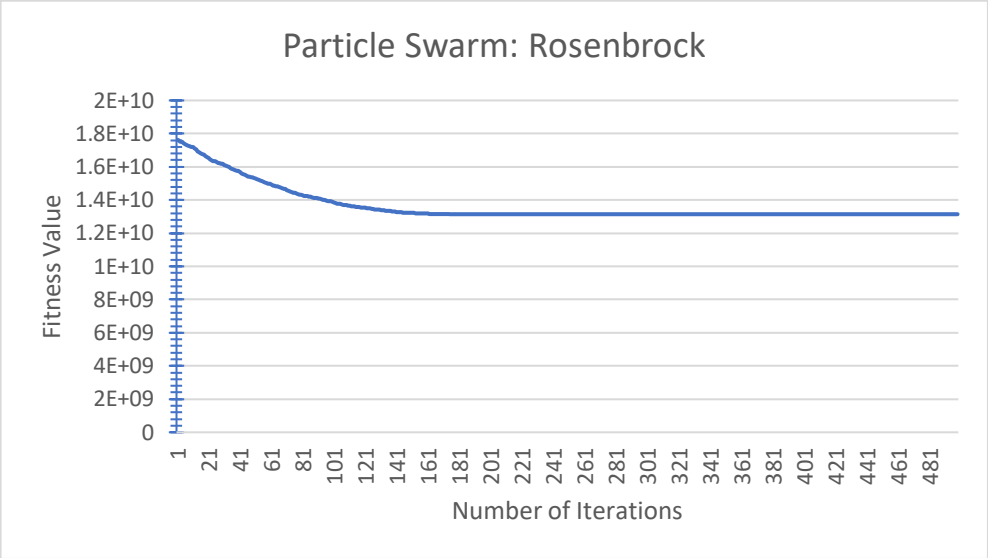

Figure 1.3: Best Fitness obtained using Particle Swarm Optimization on F(3)

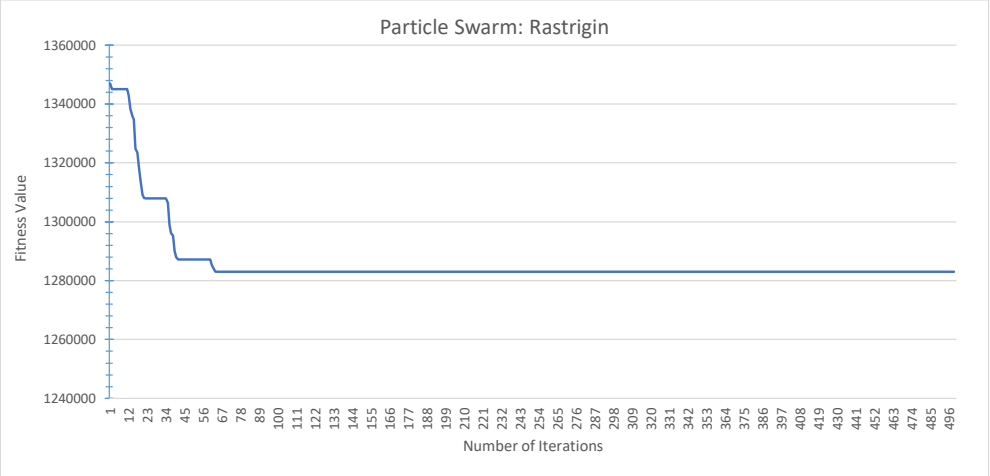

Figure 1.4: Best Fitness obtained using Particle Swarm Optimization on F(4)

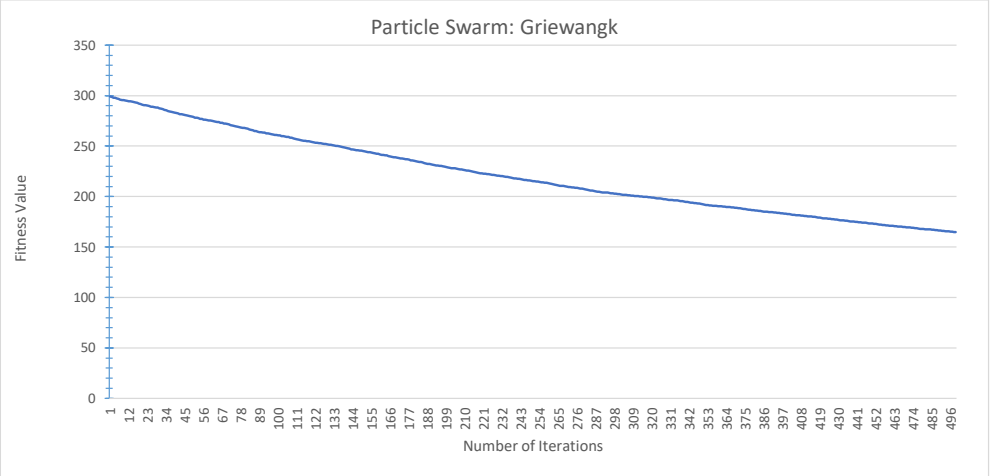

Figure 1.5: Best Fitness obtained using Particle Swarm Optimization on F(5)

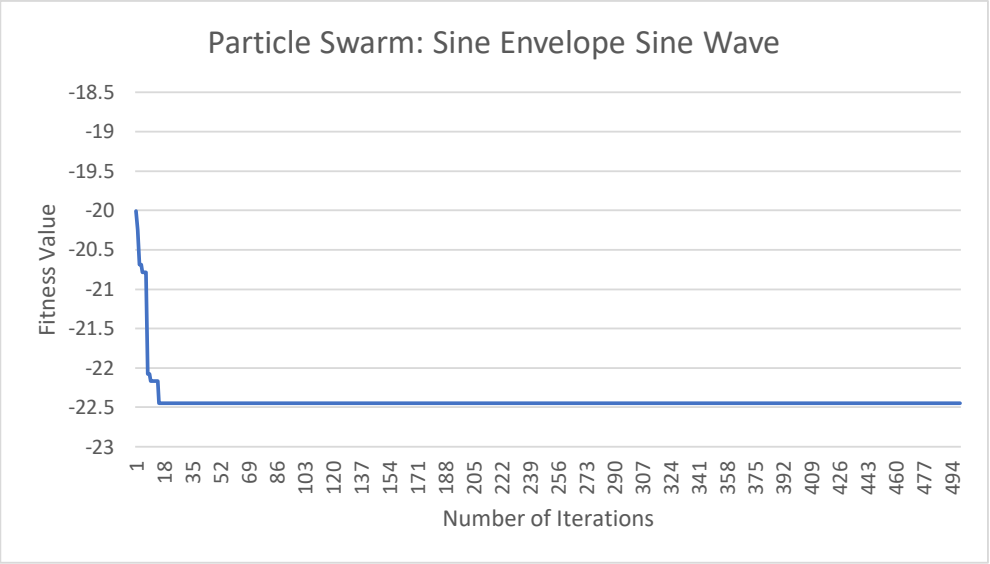

Figure 1.6: Best Fitness obtained using Particle Swarm Optimization on F(6)

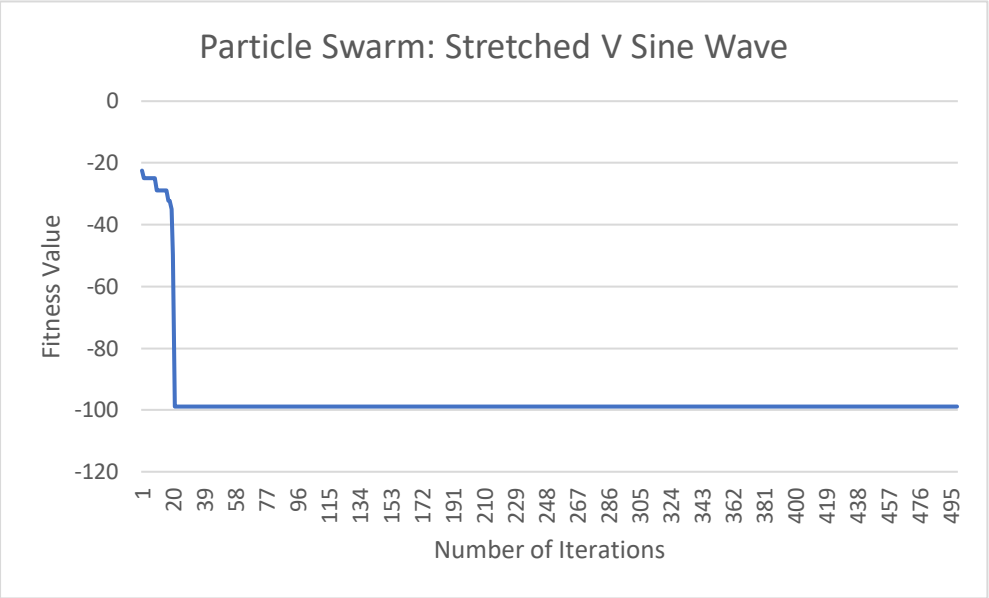

Figure 1.7: Best Fitness obtained using Particle Swarm Optimization on F(7)

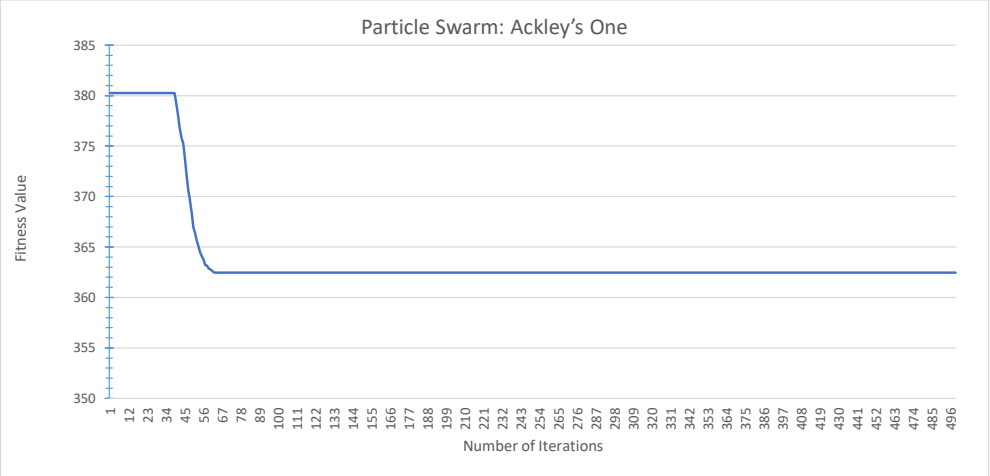

Figure 1.8: Best Fitness obtained using Particle Swarm Optimization on F(8)

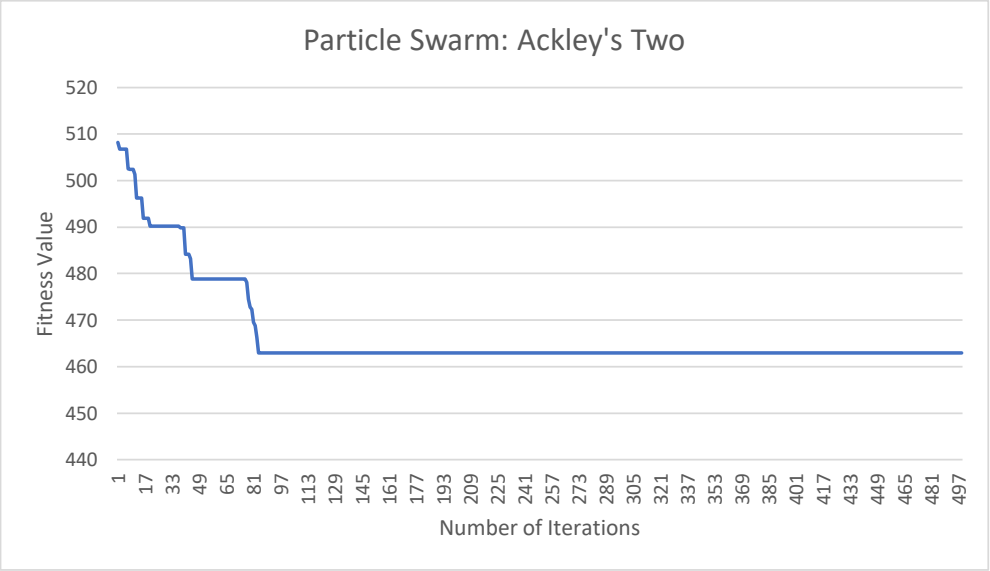

Figure 1.9: Best Fitness obtained using Particle Swarm Optimization on F(9)

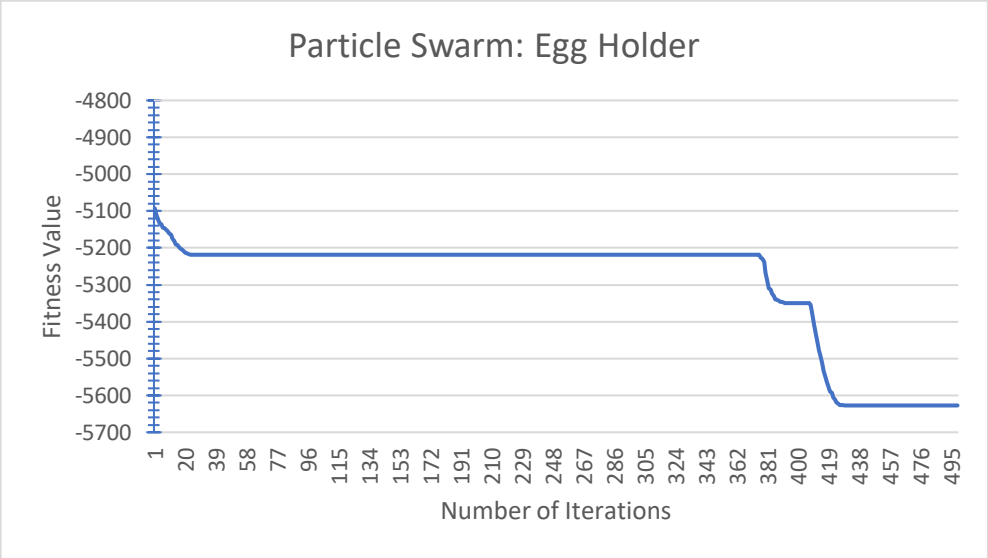

Figure 1.10: Best Fitness obtained using Particle Swarm Optimization on F(10)

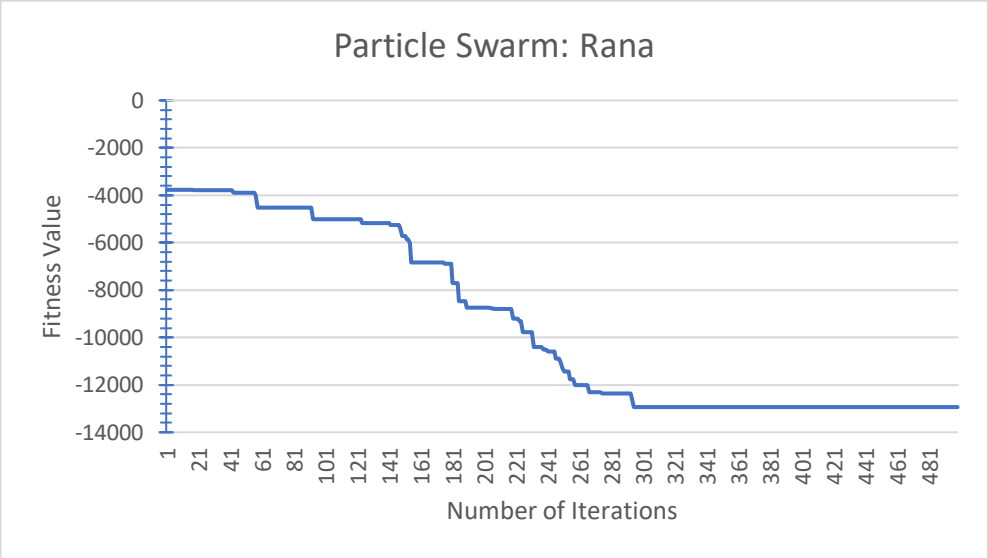

Figure 1.11: Best Fitness obtained using Particle Swarm Optimization on F(11)

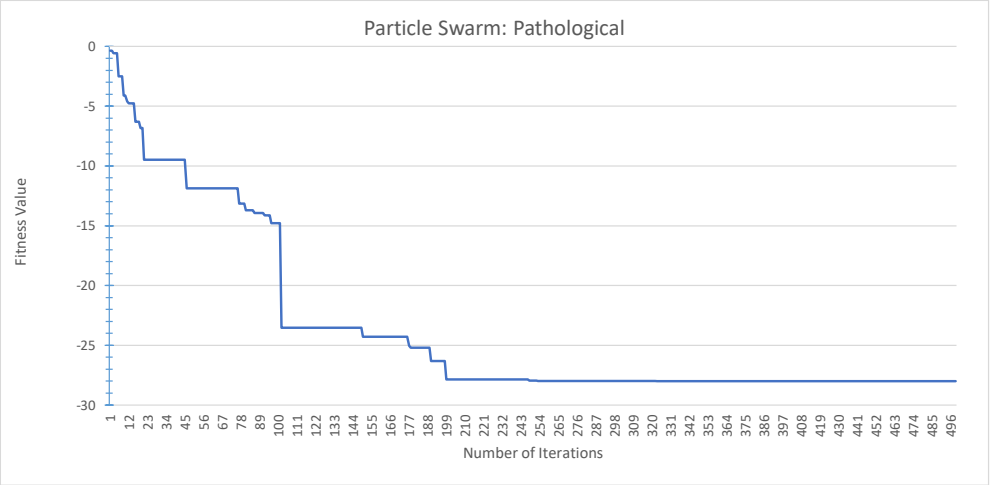

Figure 1.12: Best Fitness obtained using Particle Swarm Optimization on F(12)

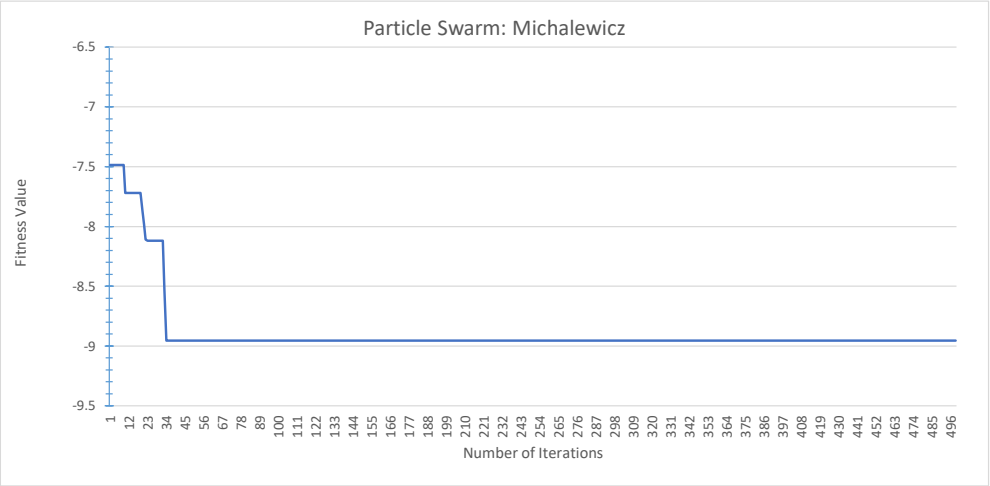

Figure 1.13: Best Fitness obtained using Particle Swarm Optimization on F(13)

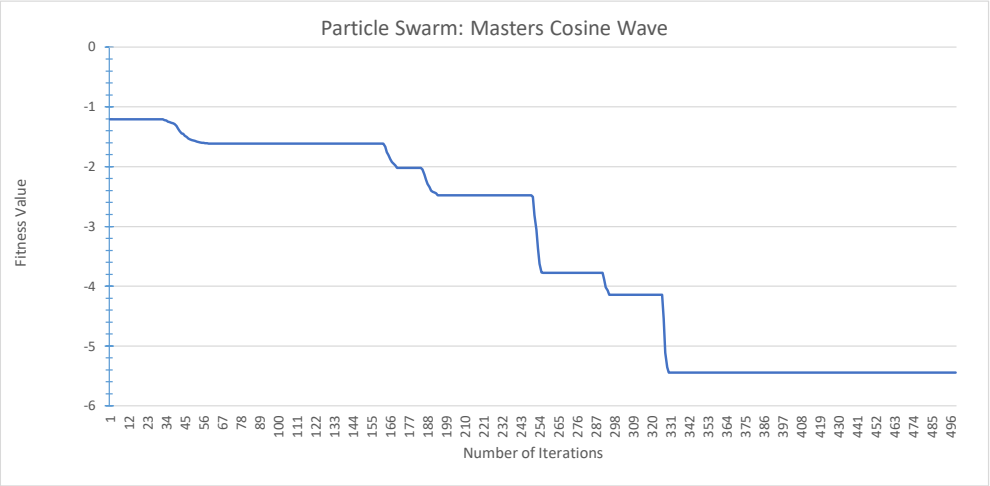

Figure 1.14: Best Fitness obtained using Particle Swarm Optimization on F(14)

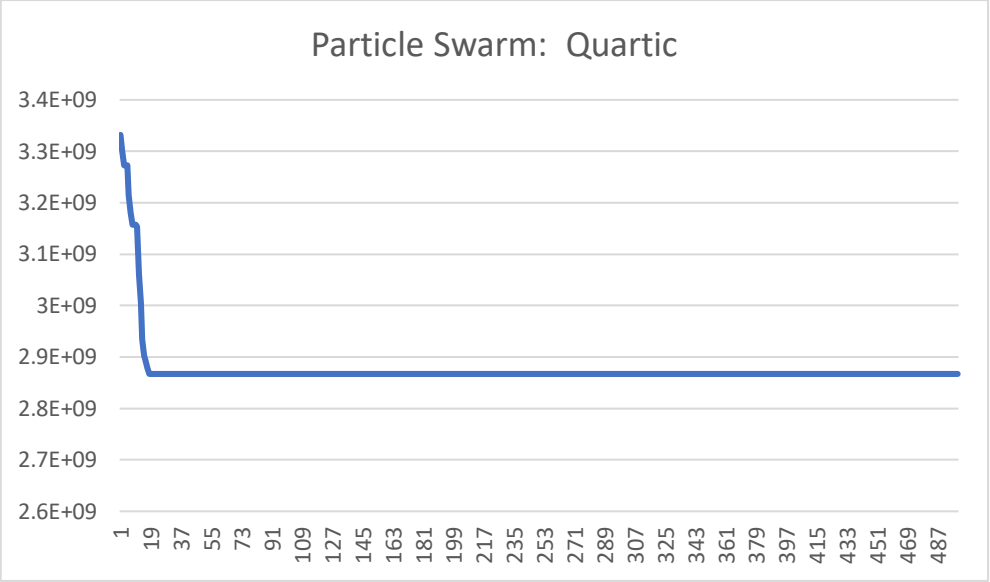

Figure 1.15: Best Worst Fitness obtained using Particle Swarm Optimization on F(15)

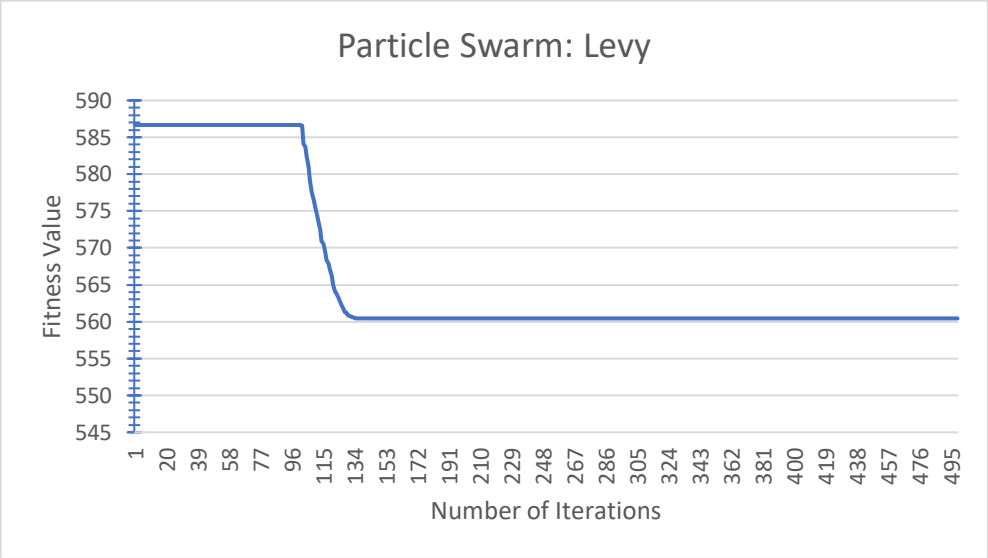

Figure 1.16: Best Worst Fitness obtained using Particle Swarm Optimization on F(16)

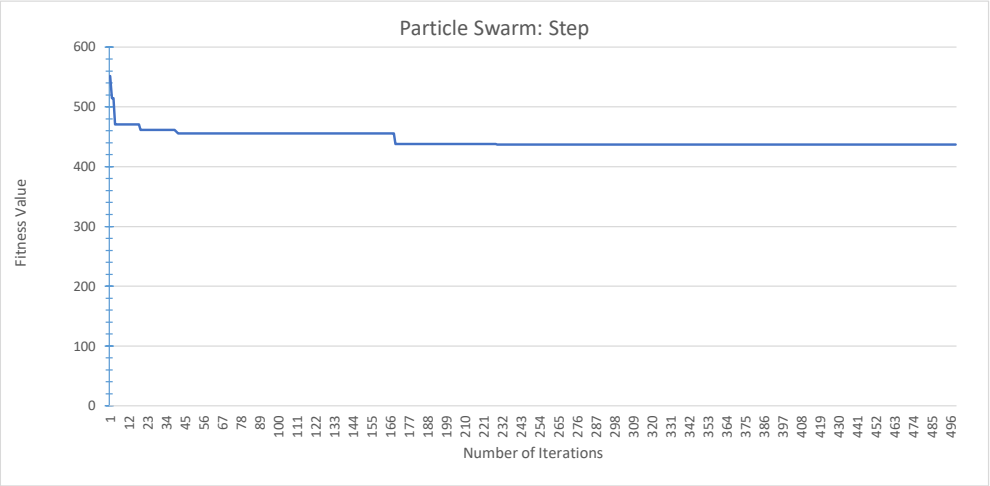

Figure 1.17: Best Worst Fitness obtained using Particle Swarm Optimization on F(17)

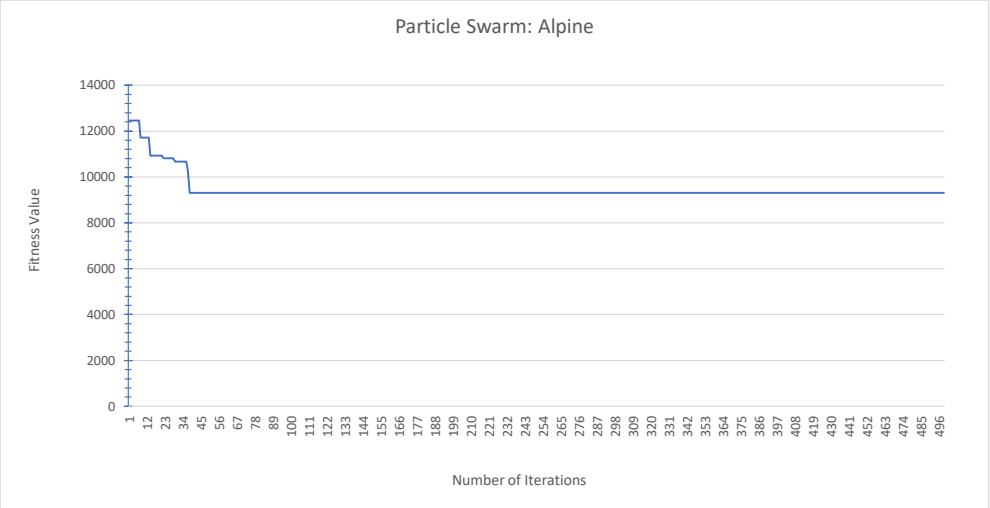

Figure 1.18: Best Worst Fitness obtained using Particle Swarm Optimization on F(18)

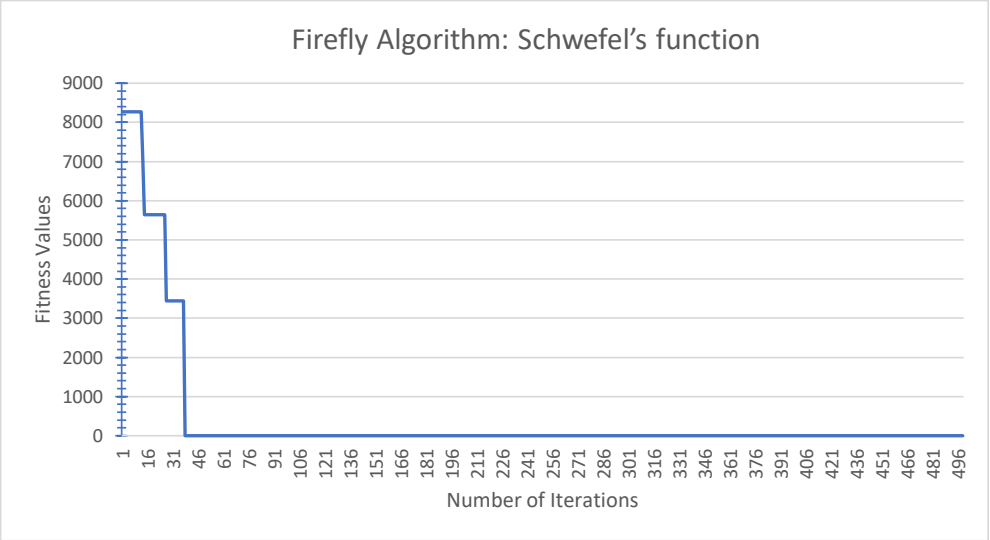

Figure 1.19: Best Fitness obtained using Firefly Algorithm on F(1)

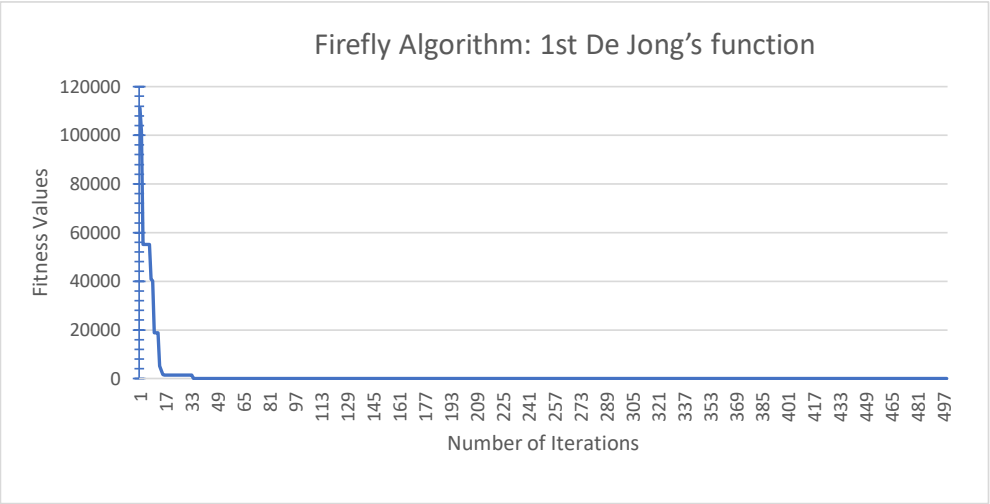

Figure 1.20: Best Fitness obtained using Firefly Algorithm on F(2)

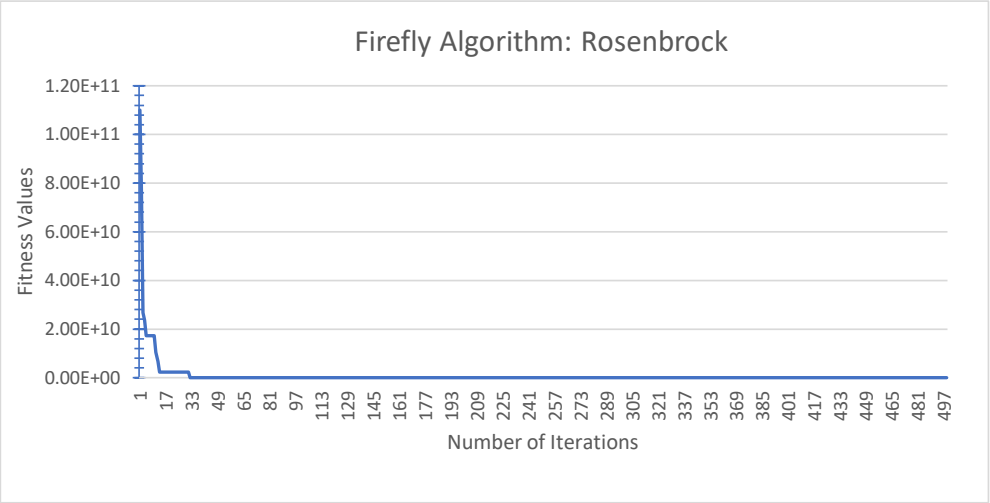

Figure 1.21: Best Fitness obtained using Firefly Algorithm on F(3)

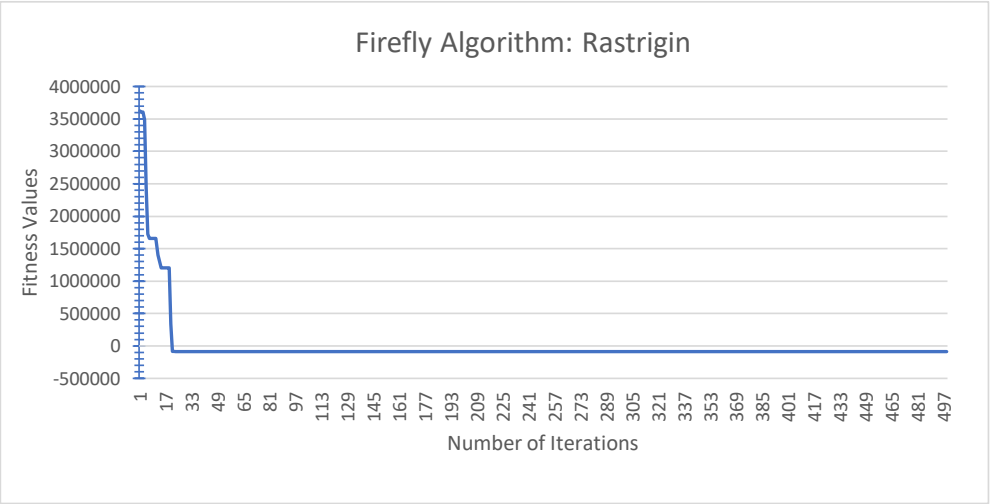

Figure 1.22: Best Fitness obtained using Firefly Algorithm on F(4)

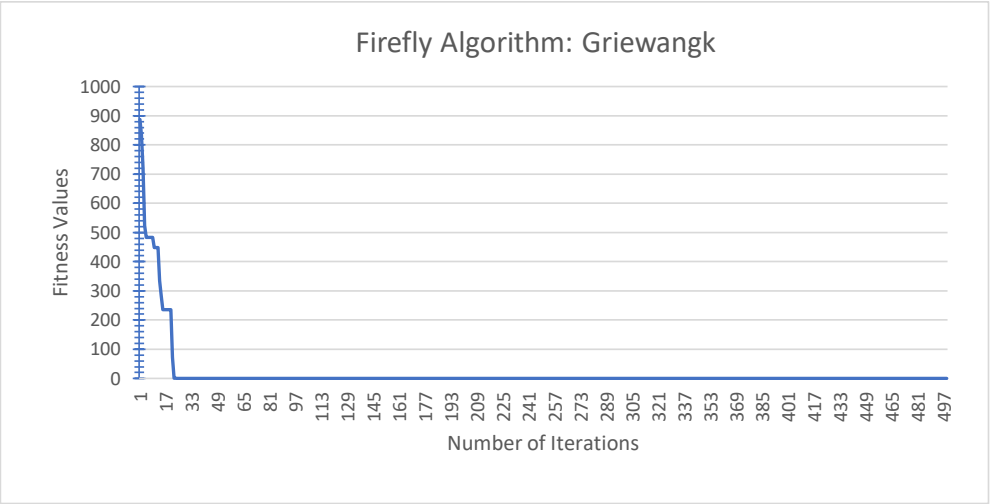

Figure 1.23: Best Fitness obtained using Firefly Algorithm on F(5)

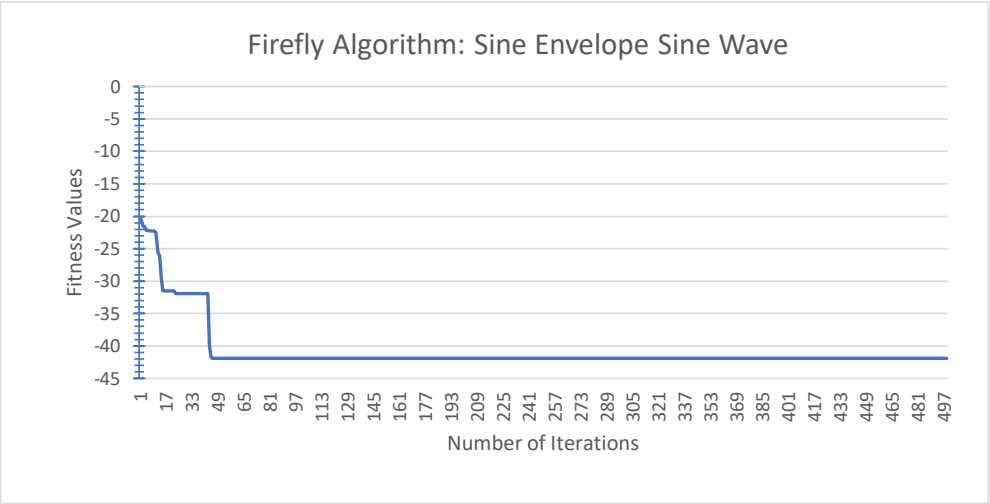

Figure 1.24: Best Fitness obtained using Firefly Algorithm on F(6)

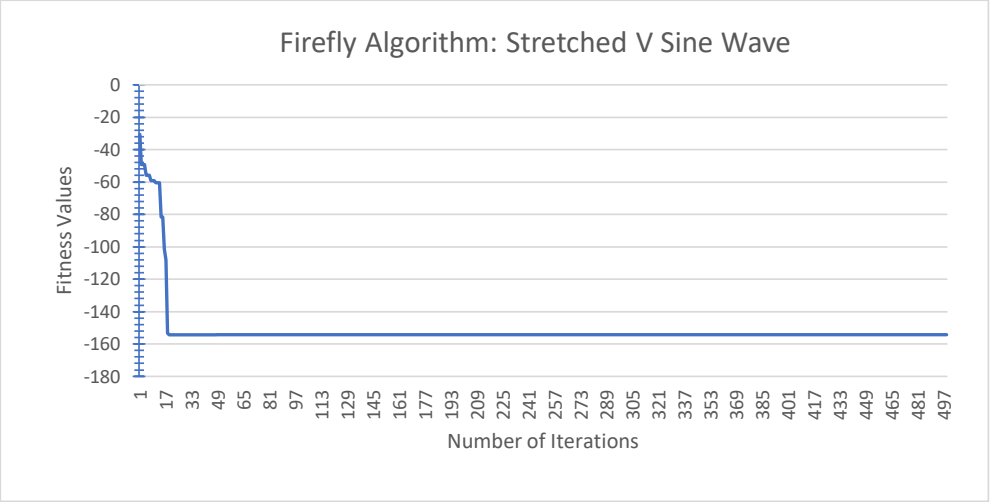

Figure 1.25: Best Fitness obtained using Firefly Algorithm on F(7)

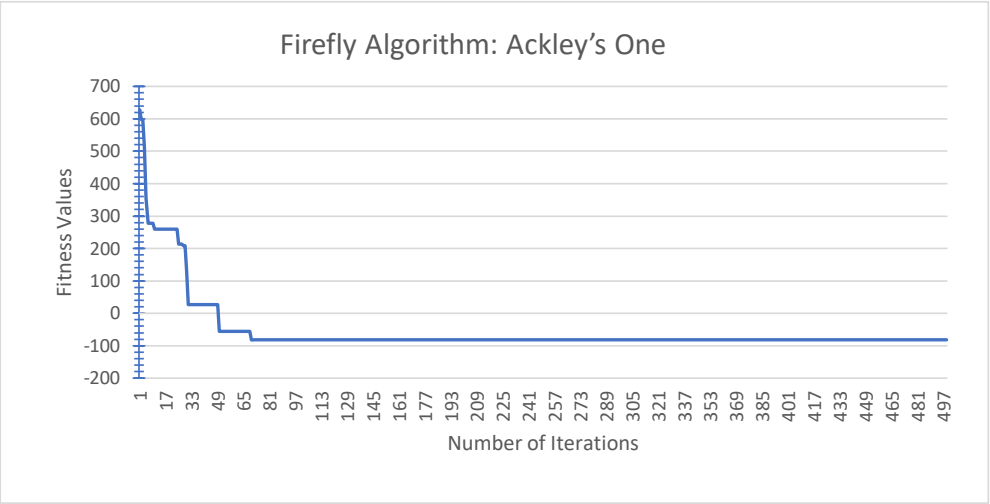

Figure 1.26: Best Fitness obtained using Firefly Algorithm on F(8)

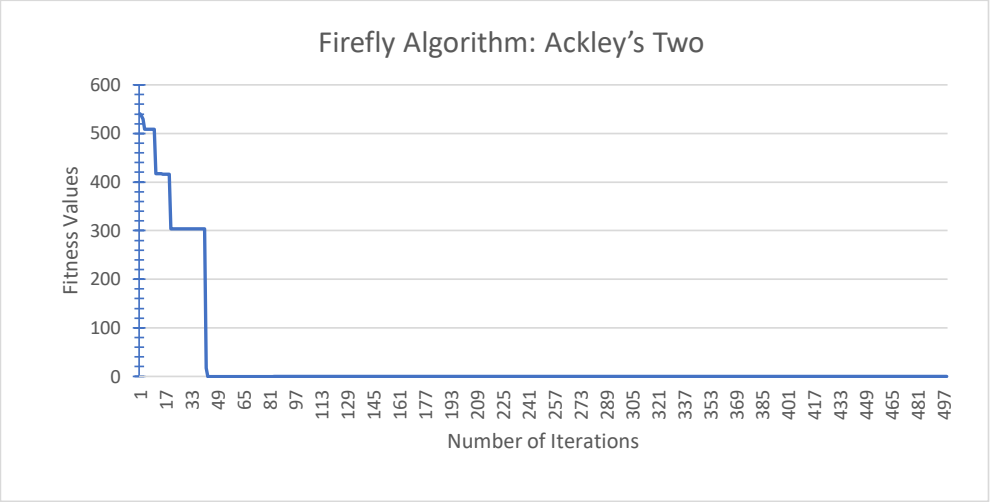

Figure 1.27: Best Fitness obtained using Firefly Algorithm on F(9)

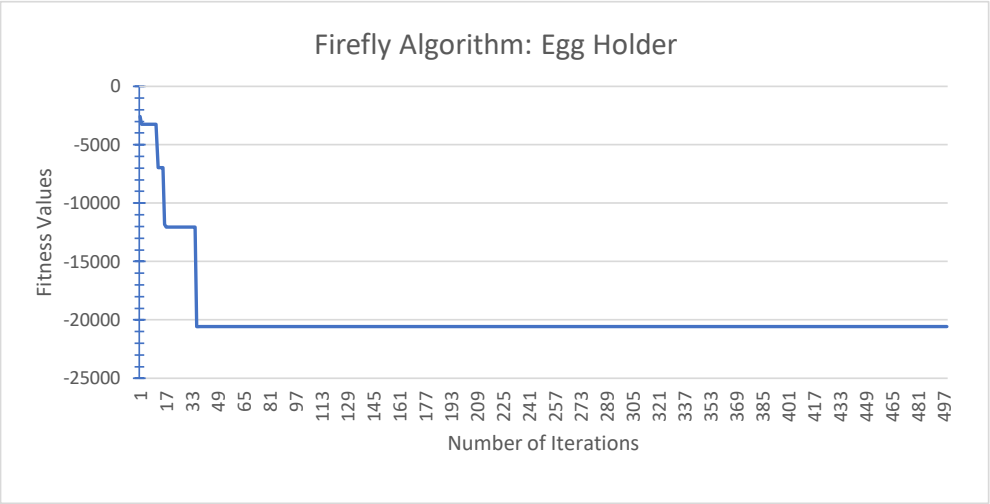

Figure 1.28: Best Fitness obtained using Firefly Algorithm on F(10)

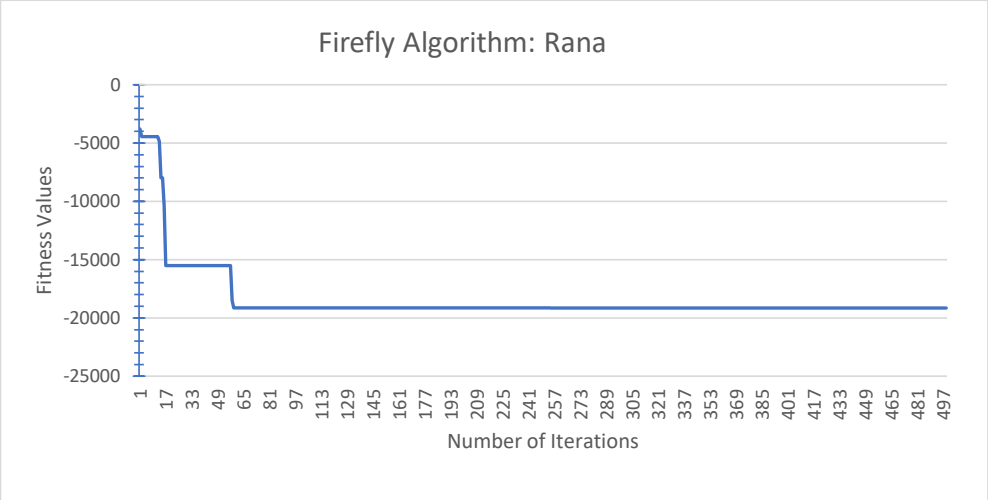

Figure 1.29: Best Fitness obtained using Firefly Algorithm on F(11)

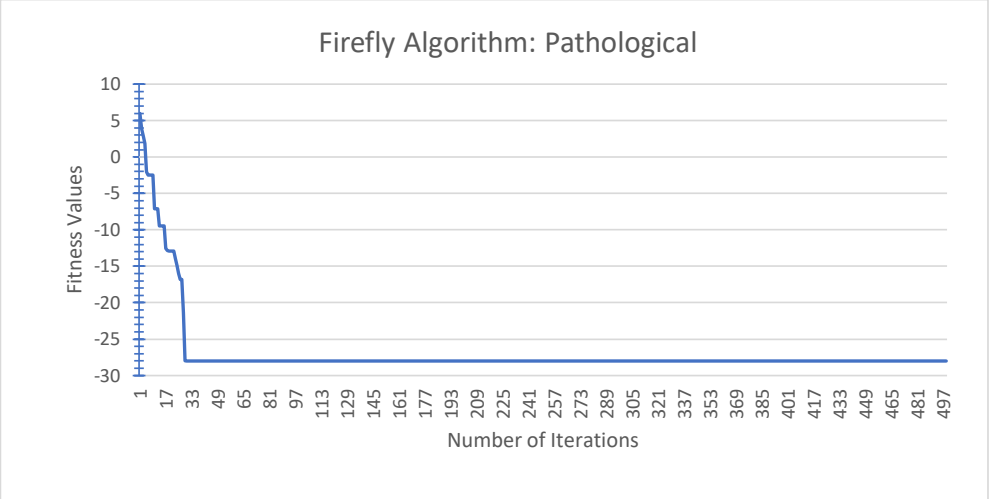

Figure 1.30: Best Fitness obtained using Firefly Algorithm on F(12)

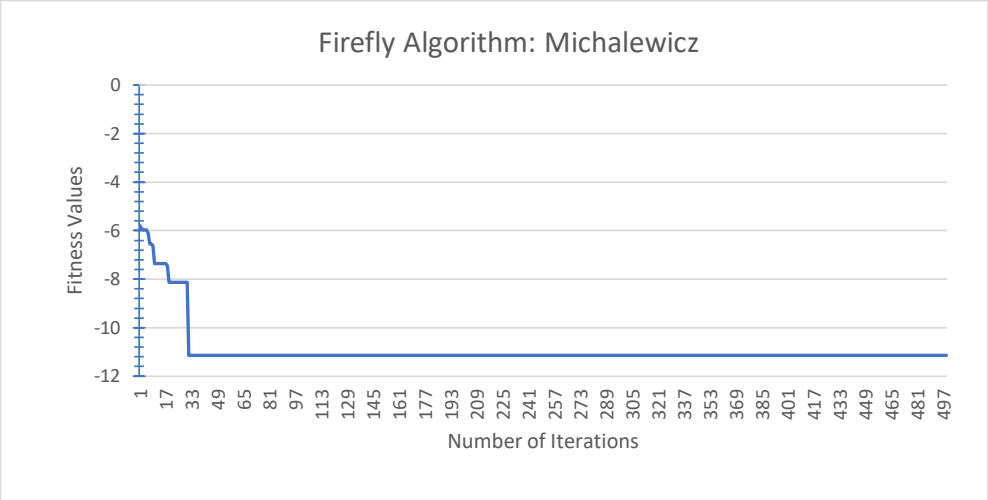

Figure 1.31: Best Fitness obtained using Firefly Algorithm on F(13)

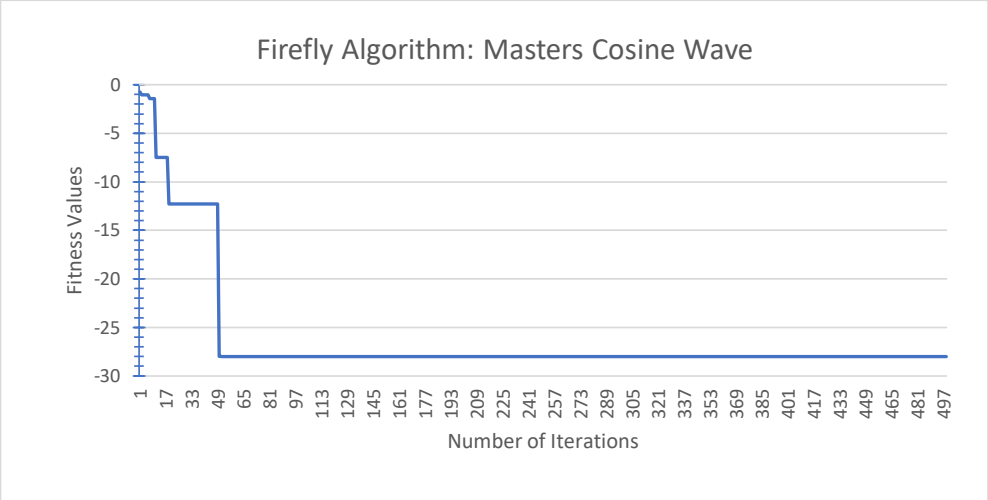

Figure 1.32: Best Fitness obtained using Firefly Algorithm on F(14)

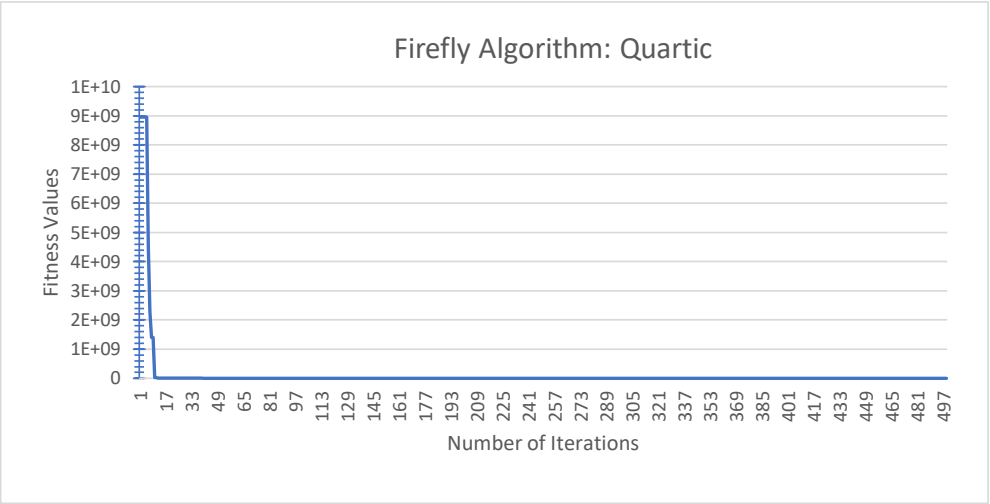

Figure 1.33: Best Fitness obtained using Firefly Algorithm on F(15)

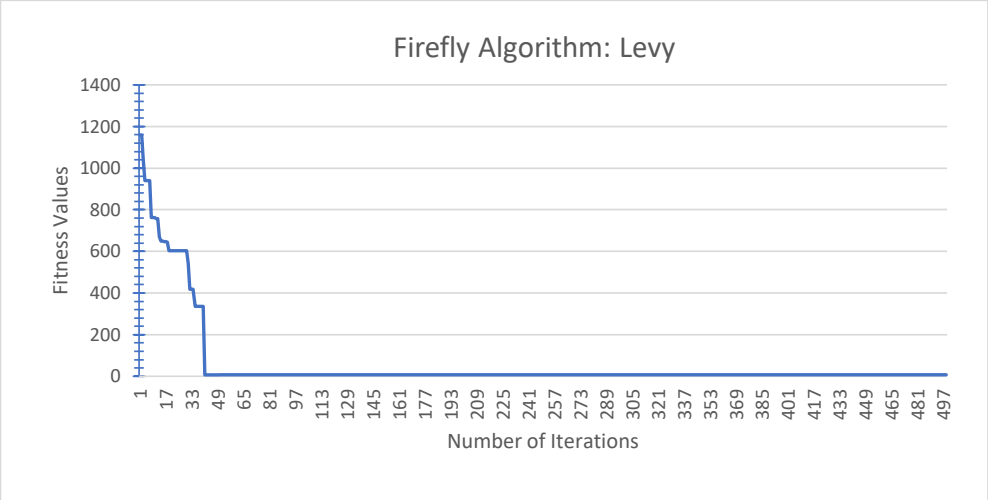

Figure 1.34: Best Fitness obtained using Firefly Algorithm on F(16)

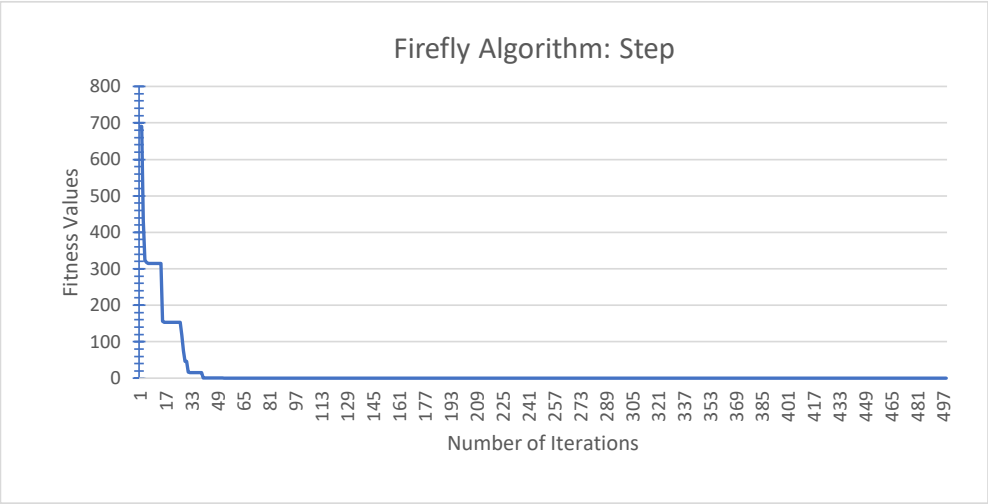

Figure 1.35: Best Fitness obtained using Firefly Algorithm on F(17)

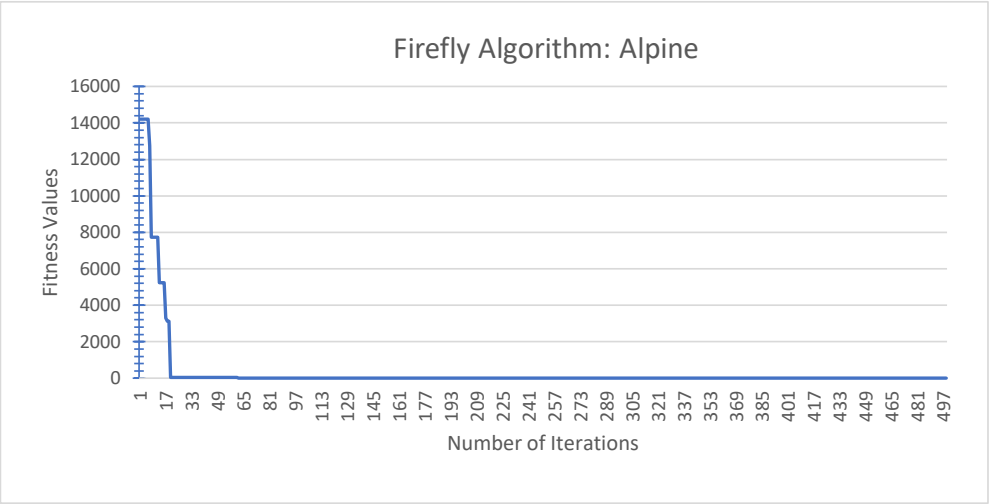

Figure 1.36: Best Fitness obtained using Firefly Algorithm on F(18)

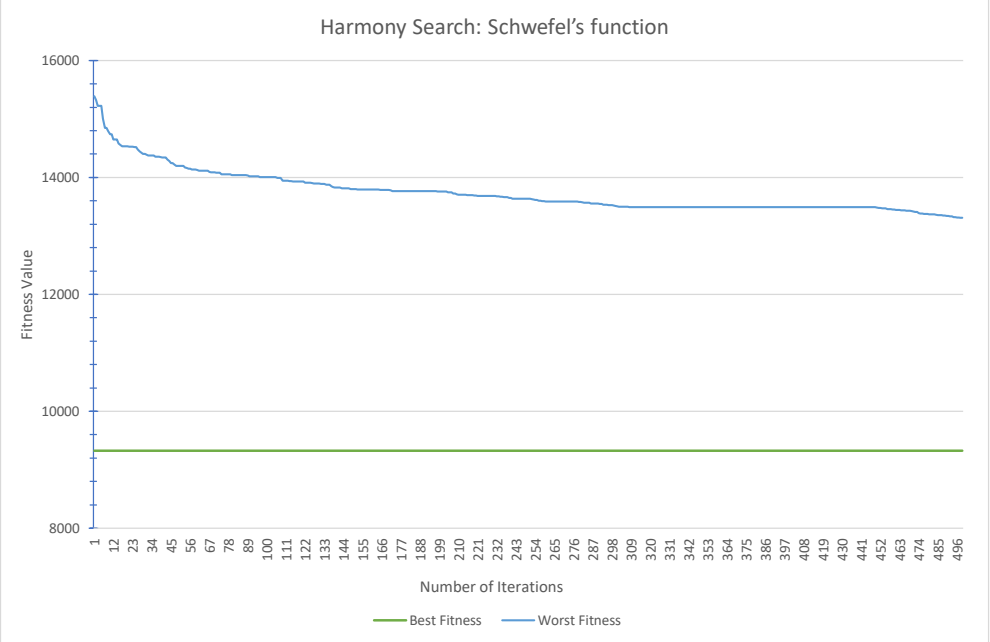

Figure 1.37: Best and Worst Fitness obtained using Harmony Search on F(1)

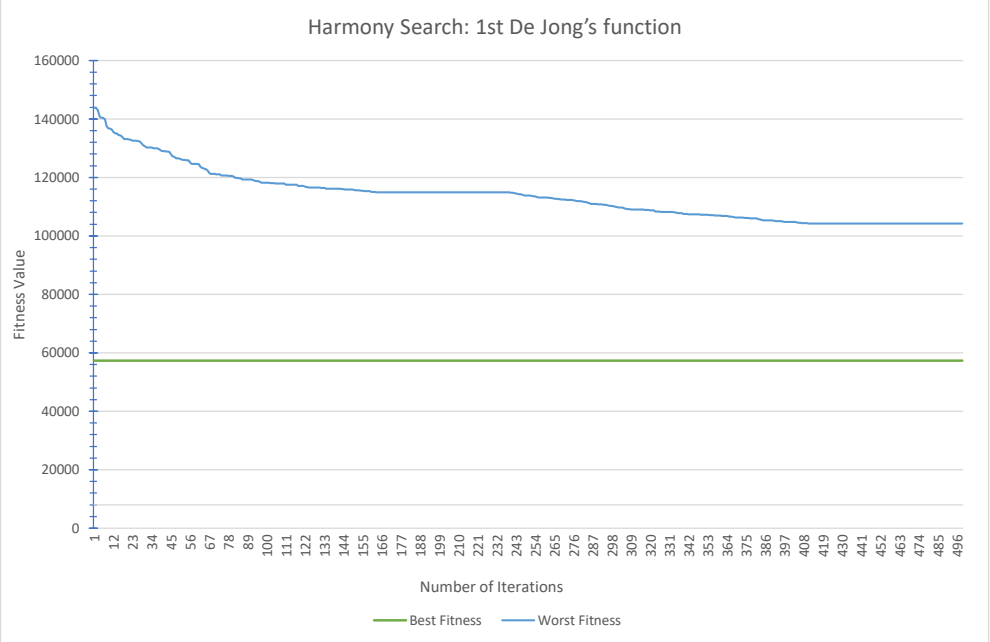

Figure 1.38: Best and Worst Fitness obtained using Harmony Search on F(2)

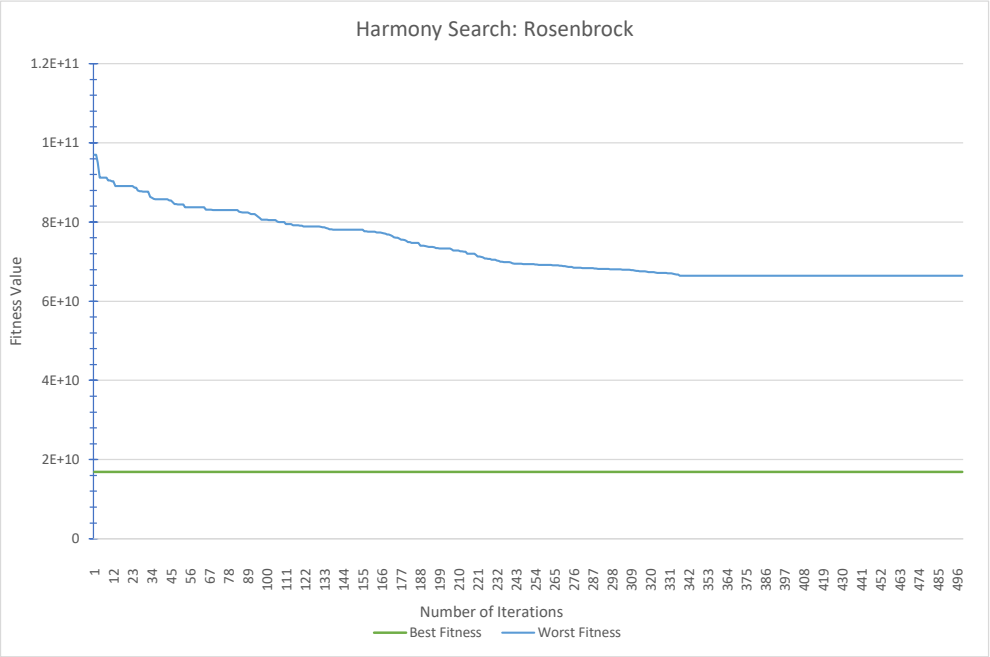

Figure 1.39: Best and Worst Fitness obtained using Harmony Search on F(3)

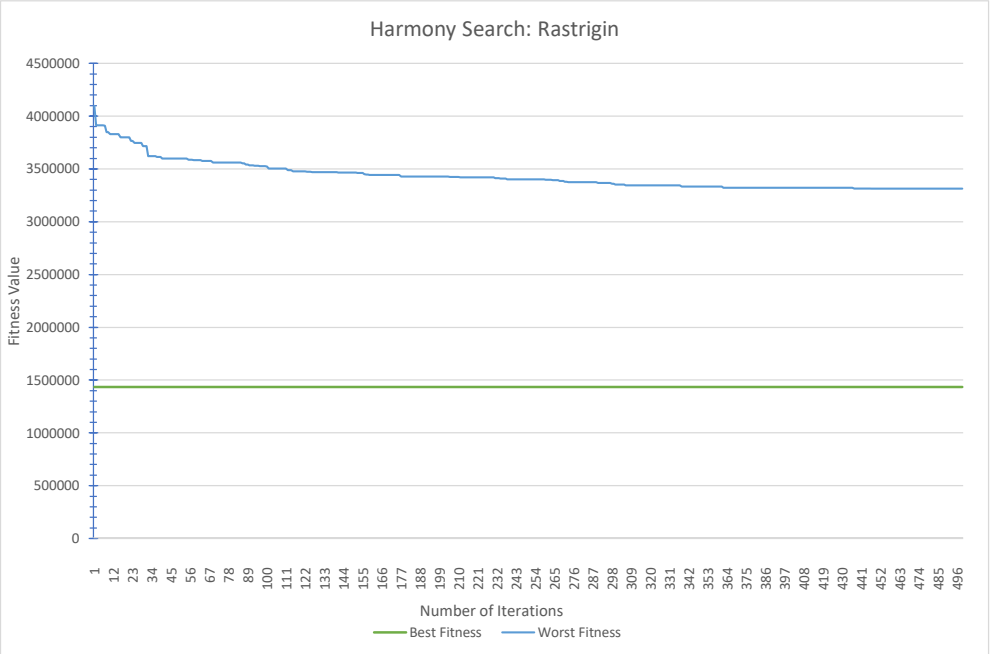

Figure 1.40: Best and Worst Fitness obtained using Harmony Search on F(4)

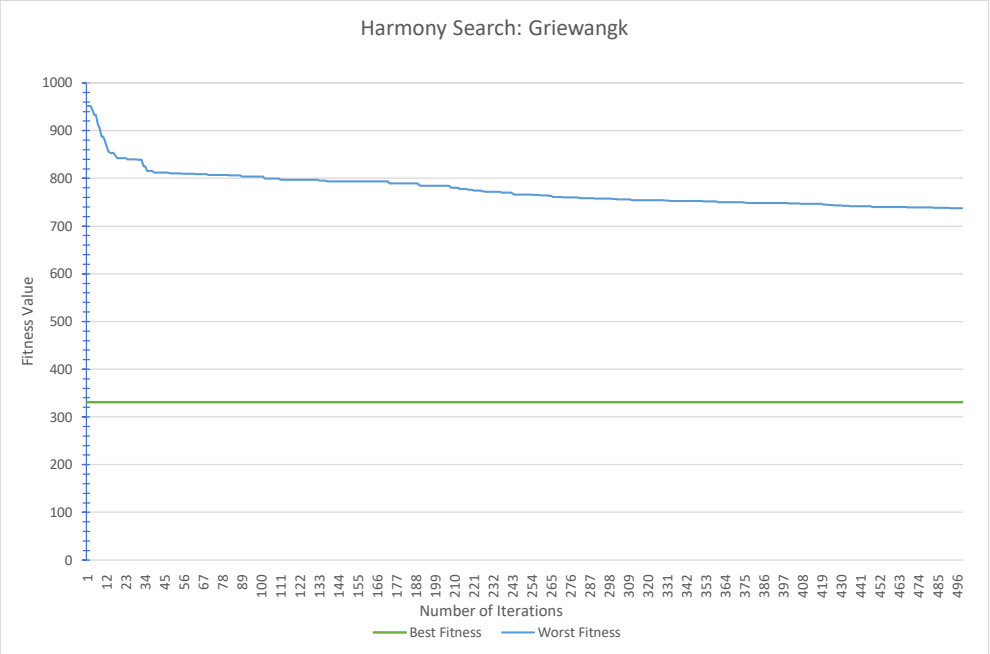

Figure 1.41: Best and Worst Fitness obtained using Harmony Search on F(5)

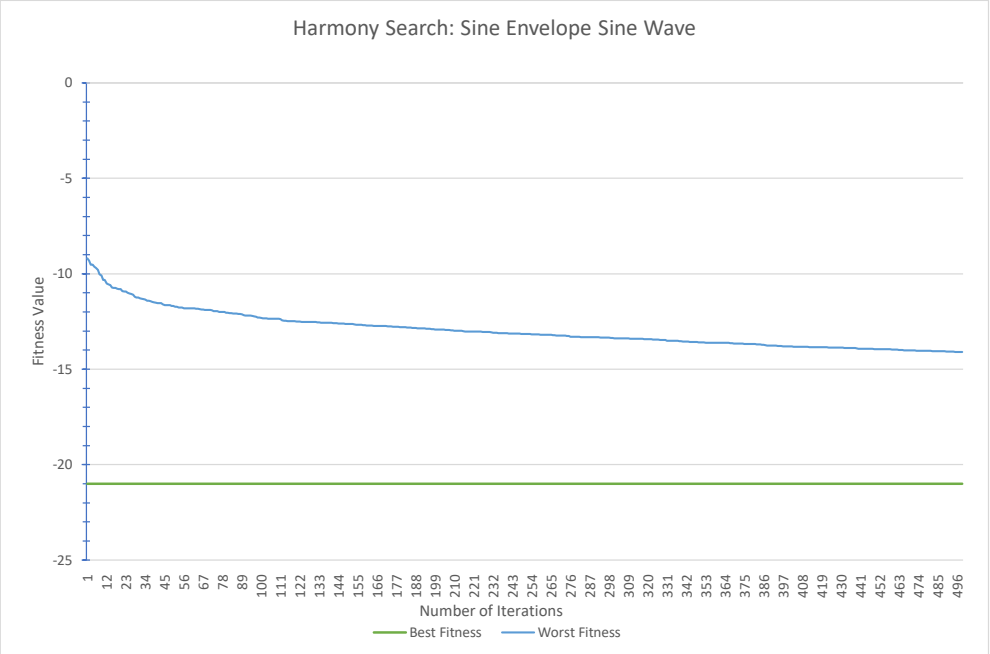

Figure 1.42: Best and Worst Fitness obtained using Harmony Search on F(6)

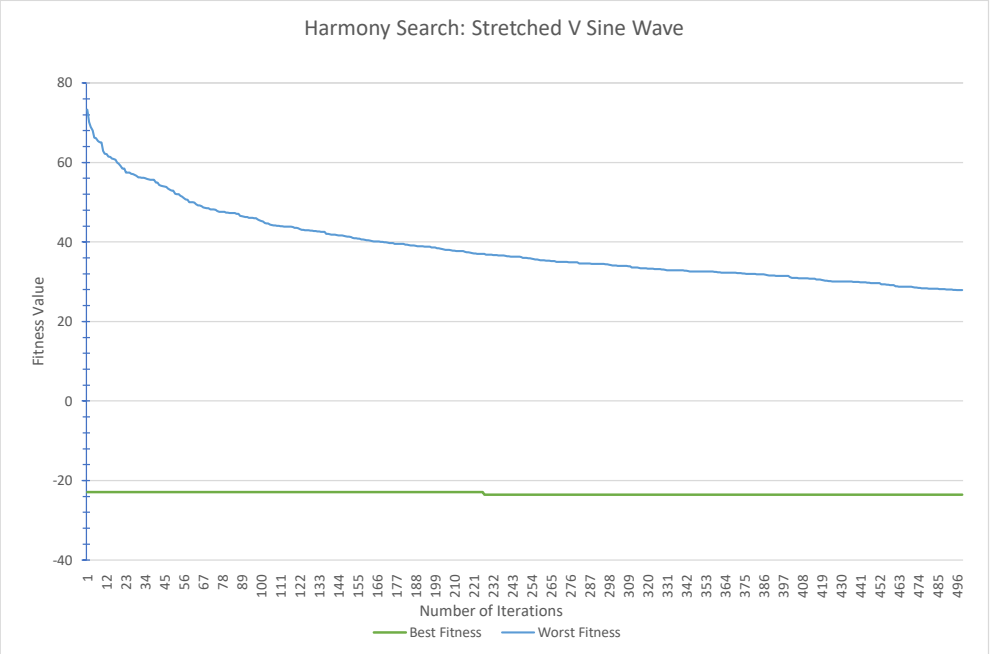

Figure 1.43: Best and Worst Fitness obtained using Harmony Search on F(7)

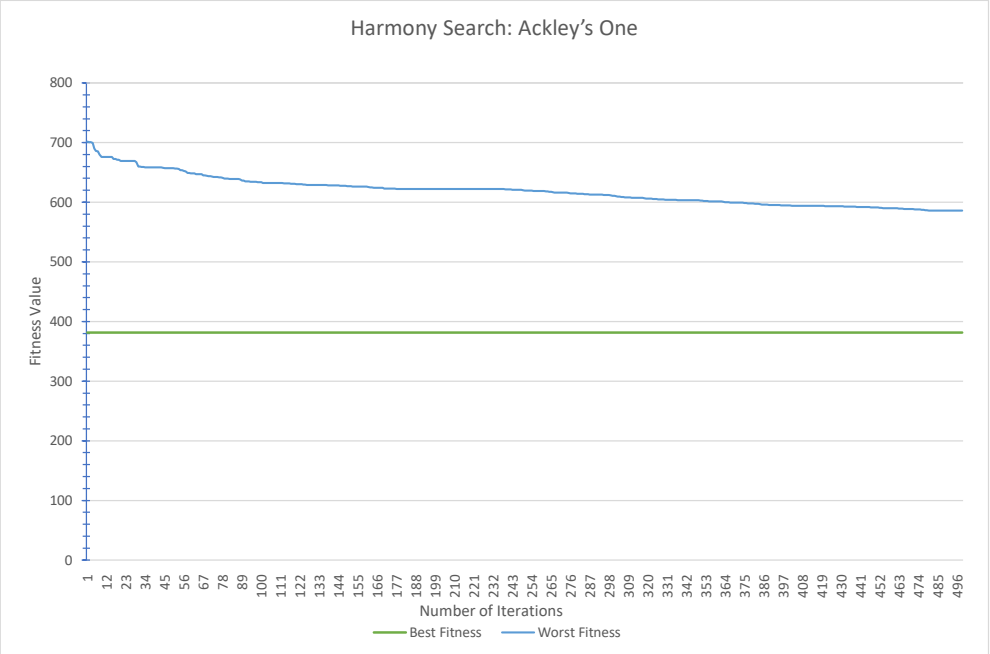

Figure 1.44: Best and Worst Fitness obtained using Harmony Search on F(8)

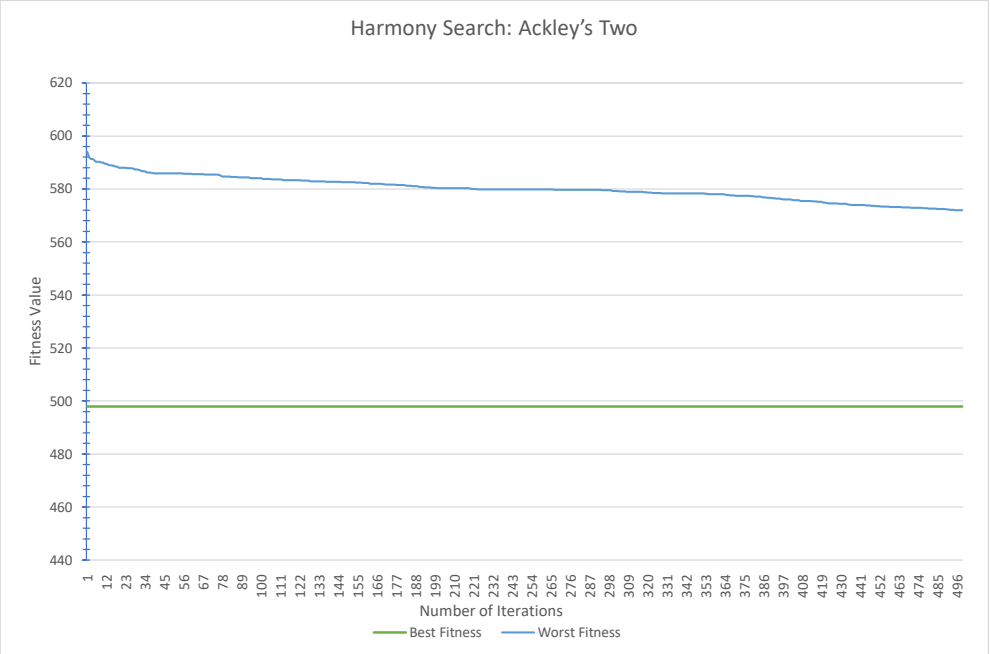

Figure 1.45: Best and Worst Fitness obtained using Harmony Search on F(9)

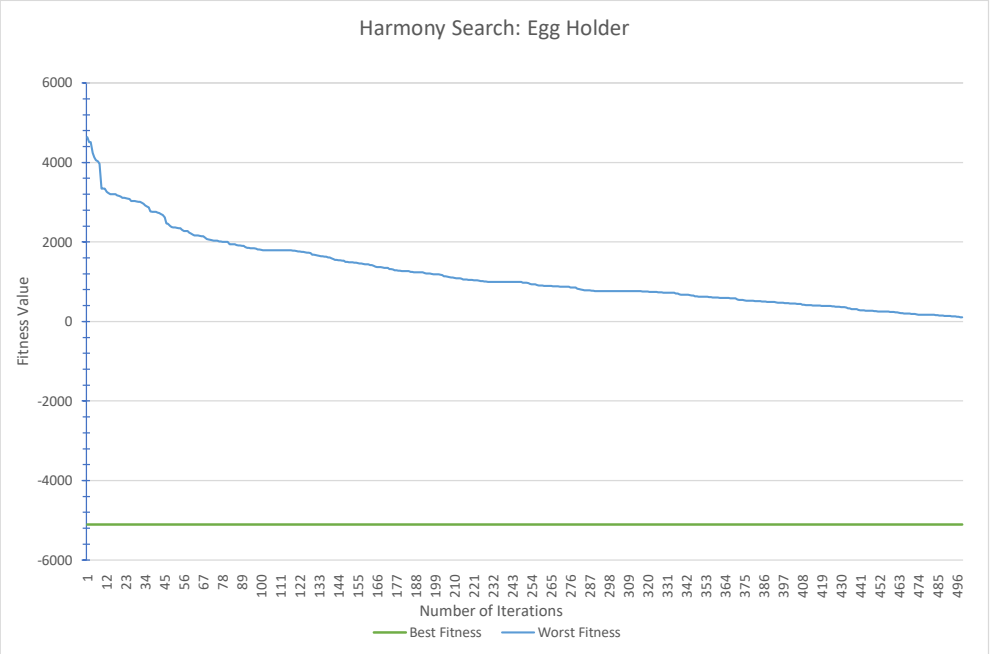

Figure 1.46: Best and Worst Fitness obtained using Harmony Search on F(10)

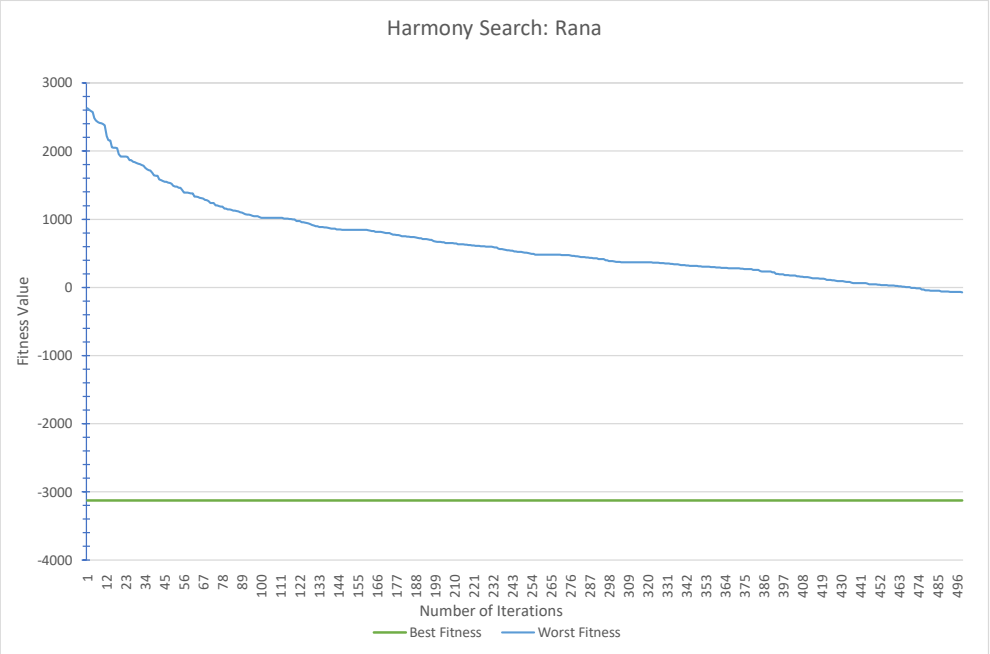

Figure 1.47: Best and Worst Fitness obtained using Harmony Search on F(11)

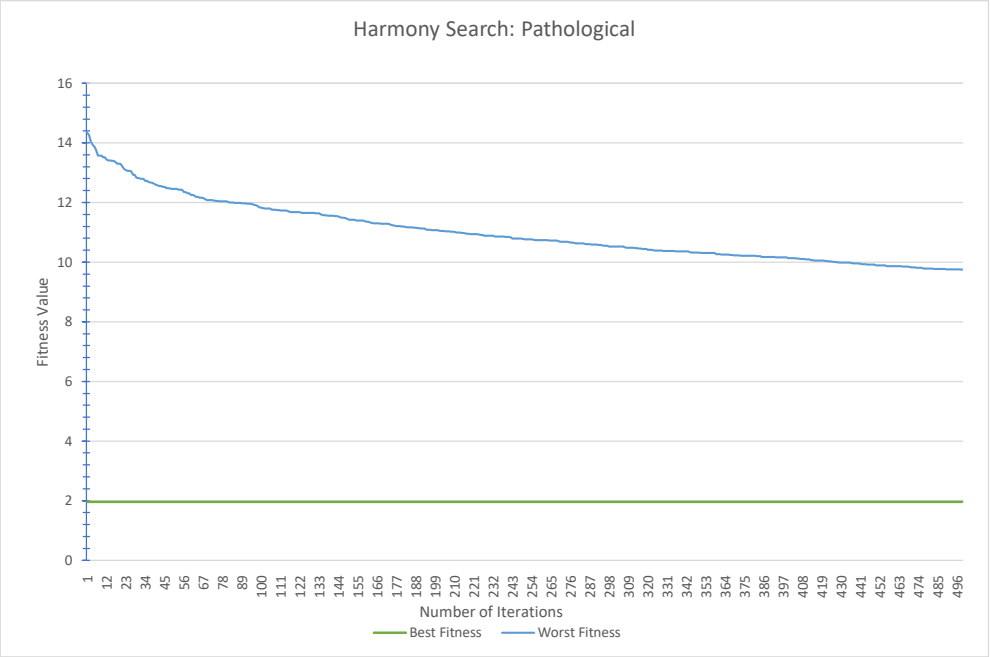

Figure 1.48: Best and Worst Fitness obtained using Harmony Search on F(12)

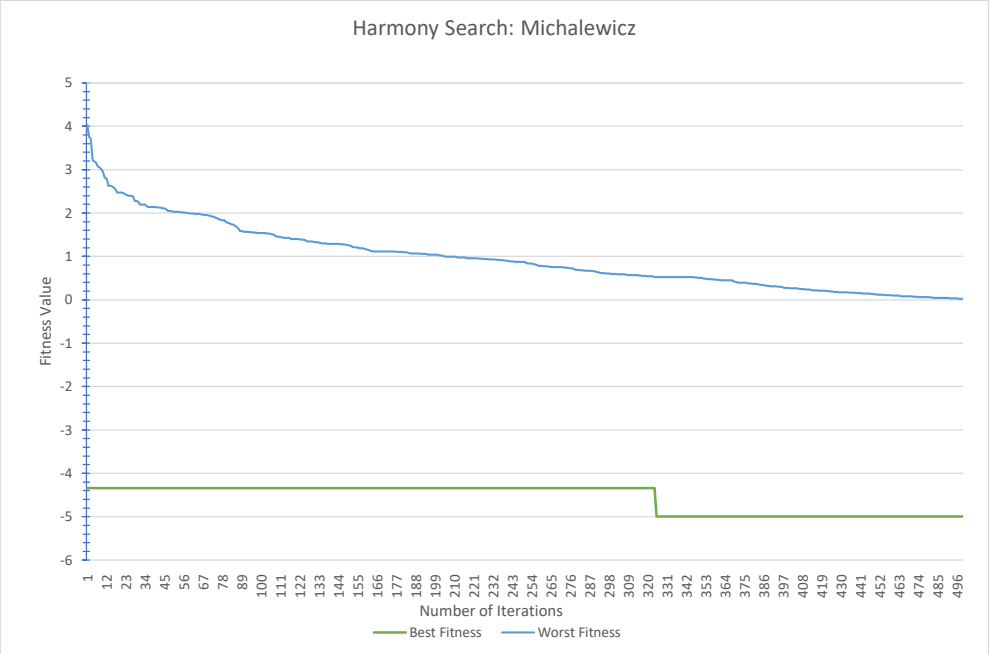

Figure 1.49: Best and Worst Fitness obtained using Harmony Search on F(13)

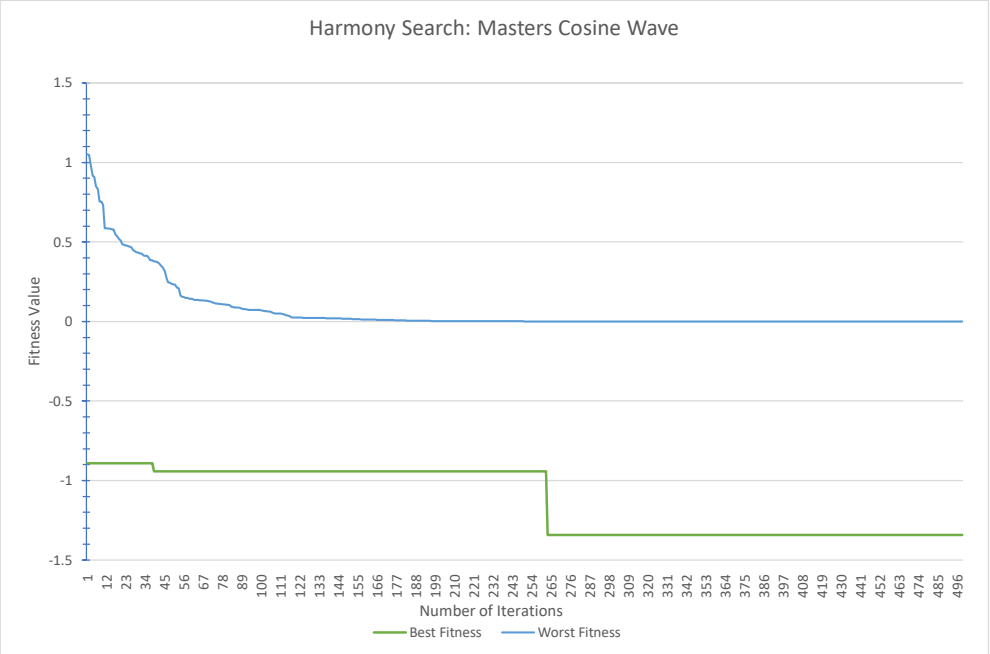

Figure 1.50: Best and Worst Fitness obtained using Harmony Search on F(14)

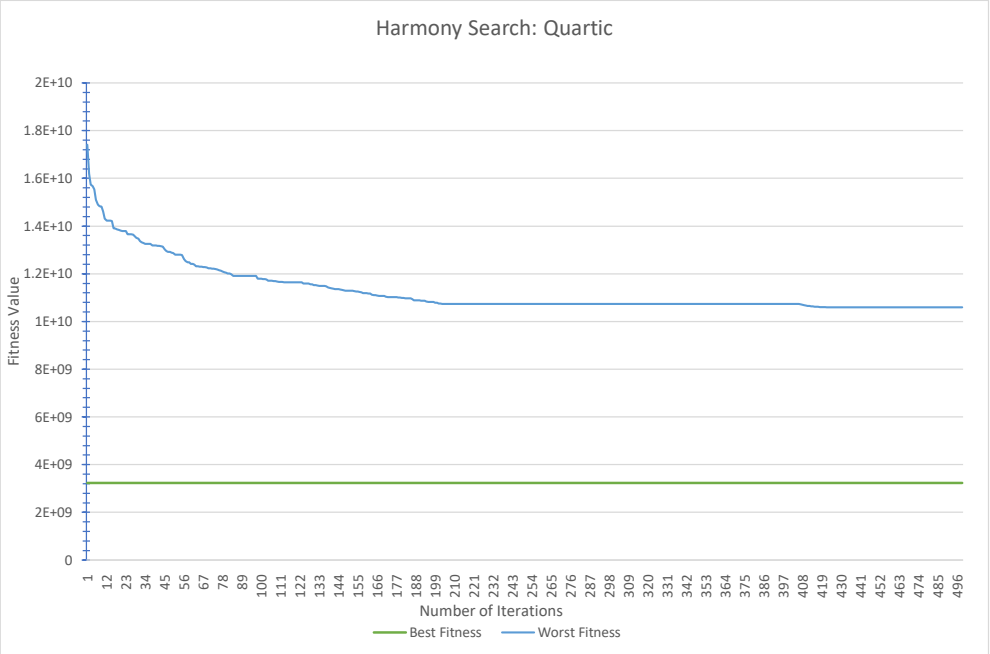

Figure 1.51: Best and Worst Fitness obtained using Harmony Search on F(15)

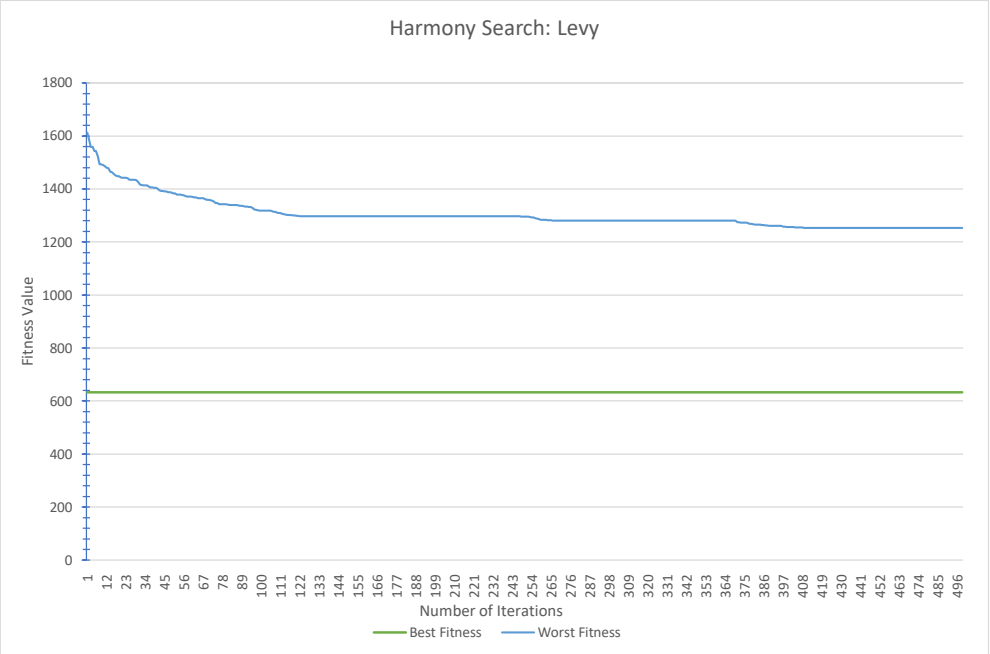

Figure 1.52: Best and Worst Fitness obtained using Harmony Search on F(16)

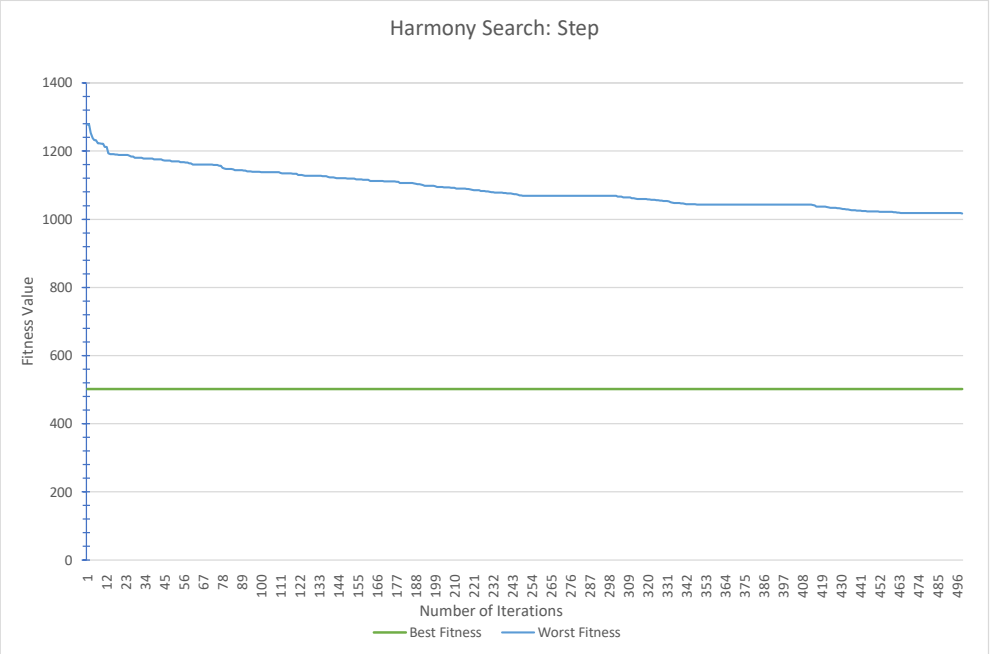

Figure 1.53: Best and Worst Fitness obtained using Harmony Search on F(17)

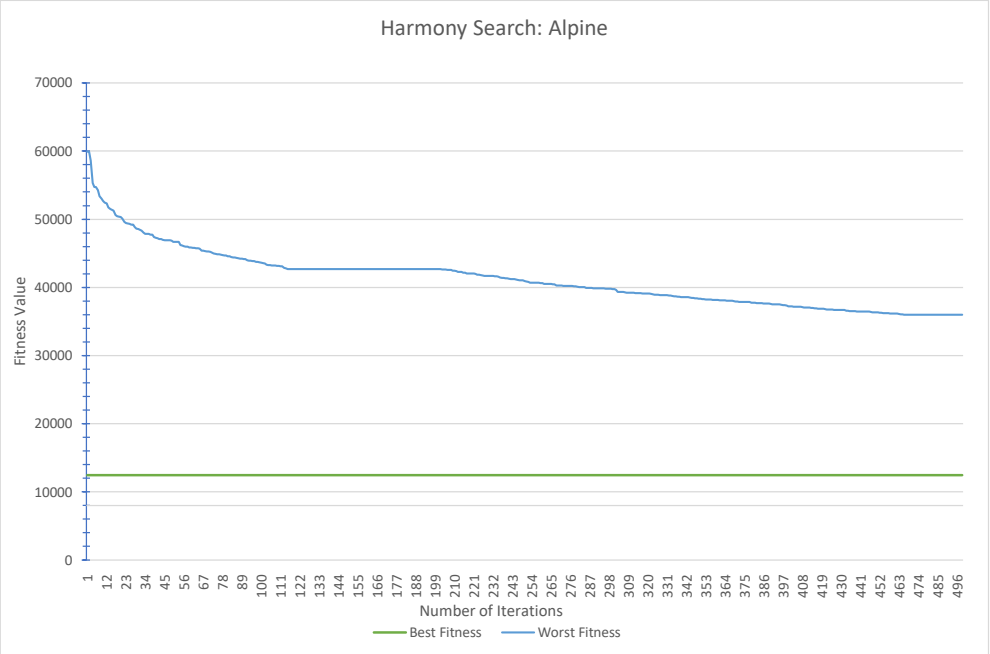

Figure 1.54: Best and Worst Fitness obtained using Harmony Search on F(18)