Central Washington University, Department of Computer Science

CS 471 Optimization - Project 4 Swarm Optimization Report

Al Timofeyev

May 20, 2019

Abstract
Project 4 - Swarm Optimization
Implement three swarm optimization algorithms, tune the parameters to achieve optimal results, and determine if a population stagnates and, if so, in what generation, for one of the Benchmark Functions.

1 Introduction

For this project, three different swarm optimization algorithms were implemented:
Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Harmony Search (HS).

Particle Swarm Optimization creates a random population using Mersenne Twister, calculates the fitness of each particle in that population, and sets the personal best fitness and solution for each particle. The global best for the whole population would be picked from one of the particles and then the PSO would begin. For each iteration of the PSO each particle's velocity is calculated based on it's personal and global best solutions, and depending on the **c1** and **c2** parameters, each particle would either move closer to it's personal best solution or it's global best solution. After each particle is moved, the global best solution of the PSO is updated.

Firefly Algorithm creates a random population using Mersenne Twister, calculates the fitness of each firefly in that population, and then the FA would begin. Each firefly is compared to every other firefly in the population. When two fireflies are compared the firefly with the smallest light intensity (worst fitness) is moved towards the brighter firefly. This continues until the FA has gone through all iterations.

Harmony Search creates a random population using Mersenne Twister, calculates the fitness of each harmony in that population, sorts everything based on fitness values, and then the HS would begin. For each iteration, only ONE new harmony is generated. If that new harmony is better than the worst harmony in the population, then the new harmony replaces the old harmony. The new harmony is a combination of harmonics from the old population; for each harmonic (dimension) in the new harmony, if a random number is less than the Harmony Memory Consideration Rate (HMCR) then a random harmonic is selected from the old population, else if a random number is less than the Pitch Adjustment Rate (PAR), then the pitch of a harmonic is randomly adjusted using the bandwidth and placed in the new harmony, otherwise, if the above two checks fail, then a new harmonic is generated randomly within the bounds of the population and placed in the new harmony. So only one new harmony is generated per generation, but it's only included in the population if it's better than the worst harmony in the population.

2 Results

For the testing of the PSO the parameters that were set were as follows:
Number of Dimensions = 30
Population Size = 500
Number of Iterations = 500
k = 0.65
c1 = 0.2
c2 = 0.8

For the testing of the FA the parameters that were set were as follows:
Number of Dimensions = 30
Population Size = 500
Number of Iterations = 500
alpha = 0.1
beta = 0.25
gamma = 1.0

For the testing of the HS the parameters that were set were as follows:
Number of Dimensions = 30
Population Size = 500
Number of Iterations = 500
HMCR = 0.95
PAR = 0.6
bandwidth = 0.3

Table 2.1: Particle Swarm Optimization Analysis For 30 Dimensions

	Particle Swarm Optimization						
Function	Average	Standard Deviation	Range (min)	Range (max)	Median	Time (ms)	Function Calls
f_1	-6.07e+23	3.75e+24	-4.77e+25	8851.34	-9.46e+09	3619.00	250500.00
f_2	1144.66	5413.83	0.00	57075.58	0.10	1743.00	250500.00
f_3	3.13e+08	1.85e+09	81.08	2.15e+10	400.88	2290.00	250500.00
f_4	1414.37	183442.63	-80882.58	1.51e+06	-60163.83	1815.00	250500.00
f_5	9.29	33.79	0.04	307.69	0.67	2636.00	250500.00
f_6	-32.48	1.42	-34.06	-25.90	-32.06	3663.00	250500.00
f_7	47.90	7.53	36.19	69.30	51.94	3652.00	250500.00
f_8	-11.45	82.62	-64.19	397.62	-47.69	4209.00	250500.00
f_9	147.63	99.65	79.84	516.94	90.55	4490.00	250500.00
f_10	-2.34e+14	1.29e+15	-1.62e+16	-5123.40	-99956.12	5613.00	250500.00
f_11	-2.04e+28	1.22e+29	-1.49e+30	-5631.13	-2.71e+14	9012.00	250500.00
f_12	12.36	0.16	12.08	13.02	12.31	5376.00	250500.00
f_13	-20.21	4.78	-24.47	-7.48	-22.68	2508.00	250500.00
f_14	-6.78	2.40	-10.36	-1.17	-6.81	4804.00	250500.00
f_15	6.21e+07	2.82e+08	0.00	2.89e+09	15.19	1672.00	250500.00
f_16	11.97	29.96	0.00	245.49	1.44	2850.00	250500.00
f_17	1334.54	5073.75	7.54	47935.87	10.70	1691.00	250500.00
f_18	77.86	103.78	3.30	534.09	23.39	1688.00	250500.00

Table 2.2: Firefly Algorithm Analysis For 30 Dimensions

	Firefly Algorithm						
Function	Average	Standard Deviation	Range (min)	Range (max)	Median	Time (ms)	Function Calls
f_1	9589.53	0.00	9589.53	9589.53	9589.53	244716.00	500.00
f_2	57058.82	0.00	57058.82	57058.82	57058.82	240585.00	500.00
f_3	1.74e+10	0.00	1.74e+10	1.74e+10	1.74e+10	235215.00	500.00
f_4	1.55e+06	0.00	1.55e+06	1.55e+06	1.55e+06	235155.00	561.00
f_5	363.73	0.00	363.73	363.73	363.73	235074.00	500.00
f_6	-24.98	0.00	-24.98	-24.98	-24.98	234649.00	500.00
f_7	63.07	0.00	63.07	63.07	63.07	234880.00	500.00
f_8	386.07	0.00	386.07	386.07	386.07	234916.00	500.00
f_9	502.65	0.00	502.65	502.65	502.65	236438.00	500.00
f_10	-4851.51	0.00	-4851.51	-4851.51	-4851.51	237939.00	500.00
f_11	-3191.86	0.00	-3191.86	-3191.86	-3191.86	244004.00	500.00
f_12	12.58	0.00	12.58	12.58	12.58	242650.00	500.00
f_13	-4.04	0.46	-4.19	-3.04	-3.56	243077.00	922579.00
f_14	-1.48	0.00	-1.48	-1.48	-1.48	234579.00	500.00
f_15	3.53e+09	0.00	3.53e+09	3.53e+09	3.53e+09	234791.00	500.00
f_16	863.16	100.09	932.99	385.34	951.19	242496.00	703547.00
f_17	60013.93	0.00	60013.93	60013.93	60013.93	234855.00	500.00
f_18	515.77	0.00	515.77	515.77	515.77	235391.00	500.00

Table 2.3: Harmony Search Analysis For 30 Dimensions

	Harmony Search						
Function	Average	Standard Deviation	Range (min)	Range (max)	Median	Time (ms)	Function Calls
f_1	9163.40	196.78	8855.81	9292.30	9292.30	97.00	1000.00
f_2	50804.19	2331.04	47985.25	55405.42	51025.46	104.00	1000.00
f_3	1.66e+10	2.50e+09	1.47e+10	2.12e+10	1.47e+10	106.00	1000.00
f_4	1.37e+06	161534.42	1.15e+06	1.49e+06	1.48e+06	89.00	1000.00
f_5	318.53	14.57	300.91	347.28	319.91	91.00	1000.00
f_6	-25.41	0.23	-26.16	-25.34	-25.34	92.00	1000.00
f_7	68.94	0.59	68.11	69.47	69.47	83.00	1000.00
f_8	382.03	14.57	340.63	389.93	385.22	100.00	1000.00
f_9	504.24	15.49	492.32	524.38	492.32	98.00	1000.00
$f_1 0$	-6107.10	471.71	-6257.25	-4625.19	-6257.25	85.00	1000.00
$f_1 1$	-3263.10	0.00	-3263.10	-3263.10	-3263.10	98.00	1000.00
$f_1 2$	13.11	0.00	13.11	13.11	13.11	87.00	1000.00
$f_1 3$	-8.24	0.48	-8.47	-7.07	-8.43	73.00	1000.00
$f_1 4$	-1.44	0.00	-1.44	-1.44	-1.44	81.00	1000.00
$f_1 5$	2.09e+09	0.00	2.09e+09	2.09e+09	2.09e+09	85.00	1000.00
$f_1 6$	190.69	9.79	182.19	201.96	182.19	75.00	1000.00
$f_1 7$	34877.10	0.00	34877.10	34877.10	34877.10	85.00	1000.00
$f_1 8$	559.74	23.15	537.75	603.06	537.75	93.00	1000.00

Table 2.4: Stagnation of Population

Dimension	Average	Standard Deviation
D_1	3574.77	5047.63
D_2	85.06	106.22
D_3	2045.50	3625.56
D_4	135.71	106.92
D_5	153.84	140.67
D_6	1899.02	1799.01
D_7	133.81	116.16
D_8	182.06	113.66
D_9	626.24	670.88
D_{10}	700.43	834.39
D_{11}	777.18	893.47
D_{12}	1337.14	1484.30
D_{13}	364.24	318.43
D_{14}	87.21	111.96
D_{15}	148.18	106.74
D_{16}	2.81e+27	1.70e+28
D_{17}	2118.23	1963.82
D_{18}	107.84	121.92
D_{19}	3030.11	4434.78
D_{20}	598.57	773.22
D_{21}	263.14	101.49
D_{22}	191.49	175.78
D_{23}	1947.58	2168.55
D_{24}	164.16	118.39
D_{25}	300.42	221.15
D_{26}	537.47	534.37
D_{27}	3734.86	4885.91
D_{28}	270.84	342.12
D_{29}	657.19	582.90
D_{30}	524.01	433.05

[1] Population Stagnation - Particle Swarm - Function 1

[2] This is from all populations of Particle Swarm - Function 1

3 Analysis

Looking at the Particle Swarm Optimization (PSO), functions 15, 16, and 18 seemed to get extremely close to their global best fitness of 0, at least in when we look at the minimum range of the fitness values. And in terms of time, function 15 was the fastest. Function 11 is still the slowest of the 18 functions, but it does, however, also produce the smallest (best) results in terms of minimization. As an additional analysis of the populations for function 1, the stagnation rate of each dimension in the populations was calculated (Table 2.4). Looking at each of the dimensions, it can be noted that dimensions 2, 4, 5, 7, 8, 14, 15, 18, 22, and 24 all stagnated or, at the very least, got very close. The table to show in which iteration the population began to stagnate is not provided in this report, but is included with the project in the Results folder.

Looking at the Firefly Algorithm (FA), it can easily be seen that it is the slowest of the three swarm algorithms. There was close to zero improvement for any of the solutions in the algorithm. For some reason, only functions 13 and 16 show any signs of fitness deviation. And looking at the function calls, it seems that only these two functions showed any improvement throughout the algorithm. Function 4 also showed signs of improvement but it is almost negligible. From all the test performed with this algorithm, it can be noted that in all those tests, only these two functions, 13 and 16, ever show any improvement. This is possibly because none of the fireflies for the other functions are ever moved, as can be seen by the number of function calls.

Looking at the Harmony Search (HS), it can be seen that functions 3 and 4 did not perform very well, both in terms of time and in the case of their average being nowhere near their global best. Functions 13 and 16 perform well in terms of time, but show very little improvement, as can be seen in the standard deviation for both functions.

4 Conclusion

The Particle Swarm Optimization was a bit slow but produced great results in minimizing the 18 functions. The Firefly Algorithm is the slowest of all three, and showed almost zero improvement for all functions except functions 4, 13, and 16. The Harmony Search was the fastest of the three but it showed minimal improvements for the 18 functions.
It can be concluded that the Particle Swarm worked best for functions 1, 10, 11, and 18, the Firefly Algorithm worked best for functions 4, 13, and 16, and the Harmony Search worked best for function 10.

5 List of Functions