Function value optimization with PSO

• Problem Description:

The problem purpose is to trying minimize function value by finding global optimum point in the search space.

benchmark functions are described as bellow:

	Name	Test Function	S	Global opt.	f_{min}
E	Rosenbrock	$f_5(\vec{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^n$	$[1]^n$	0
	Step	$f_6(\vec{x}) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100,100]^n$	$[-0.5]^n$	0
M	Ackley	$f_9(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32,32]^n$	$[0]^n$	0
	Griewank	$f_{10}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$	$[0]^n$	0
H	Rastrigin	$f_8(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12,5.12]^n$	$[0]^n$	0
	Generalized Penalized	$f_{11}(\vec{x}) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50,50]^n$	$[0]^n$	0

Description:

n denotes the solution space dimension, S denotes a subset of R^n, and the global optimal solution and the global optimal value f_{min} of classical benchmark functions are given in column 5 and column 6, respectively. Ten independent experiments must be completed for each optimization function considering $n = 10, 30, and\ 50$.

• Algorithm Description:

PSO algorithm is a decentralized Swarm Intelligence search process. The swarm consist of particles with position and velocity related to them. Each particle remembers its best point ever seen as parameter calls “**pbest**”. The whole swarm best reached point remembers as parameter calls “**gbest**”.

The basic concept of PSO lies in accelerating each particle toward its **pbest** and the **gbest** locations, with a random weighted acceleration at each time.

```
Randomly generate an initial population
repeat
    for i = 1 to population_size do
        if f(x_i) < f(p_i) then p_i = x_i;
        p_g = min(p_neighbours);
        for d =1 to dimensions do
            velocity_update();
            position_update();
        end
    end
until termination criterion is met.
```

Our PSO algorithm properties come in below:

- **Initializing:**

  Initial particles position set randomly base on problem domain.
  initial particles velocity takes positive and negative 10% of particles position as velocity.

- **Position & velocity update:**

$$v_{id}^{new} = w_i \cdot v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id})$$
$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

  d is the dimension, c_1 and c_2 are positive constants, $rand_1$ and $rand_2$ are random numbers, and w is the inertia weight.
  usually $c_{1}+ c_{2} = 4$. No good reason other than empiricism.

- **Pbest & Gbest update:**

$$Pbest_i(t+1) = \begin{cases} Pbest_i(t) & if \;\; f(x_i(t+1)) \geq Pbest_i(t) \\ x_i(t+1) & otherwise \end{cases}$$

$$Gbest(t+1) = argmin_{Pbest_i} f(Pbest_i(t+1))$$

- **Inertia weight update:**

  - Large inertia weight facilitates global exploration
  - small on facilitates local exploitation

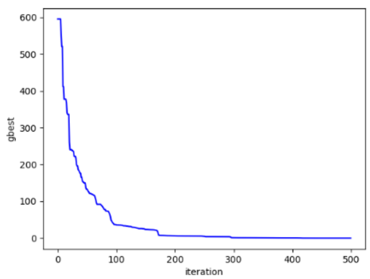
  By decreasing the inertia weight best performance archives.
  many research works are conducted where the value is chosen as: w(initial) = 0.9 and decrease to w(final) = 0.2.

  We use exponential manner for this purpose as below:
  For initial iteration we set w= 0.9 and for the rest we keep it as 0.2.

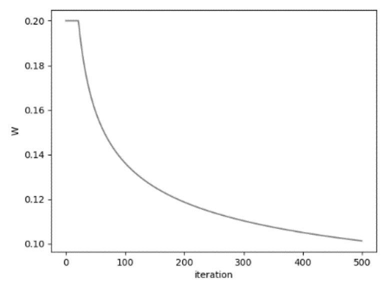
- PSO learning process:

N = 30
ITERATIONS = 500
SWARM_SIZE = 80
wMax = 0.9
wMin = 0.2
inertia mode = logarithmic

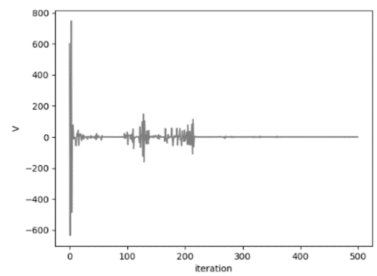
- **Griewank**



Gbest /iteration



inertia weight /iteration



velocity/iteration

- PSO Results without bounded velocity:

Rosenbrock							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0/183	36/113	246/425	73/68	3/534	3/724	3/925
30	36/27	91264283	2/42E+08	1/12E+08	30/413	31/808	34/178
50	372345193	427387669	5/65E+08	52384448	81/514	85/15	88/959
Step							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0	0	0	0	1/532	1/764	2/091
30	0	18830/998	73022/25	29188/32	10/902	11/979	13/843
50	3/867	106877/16	129563/6	36214/94	29/053	31/056	32/958
Ackley							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0	0/116	1/155	0/347	3/071	3/155	3/31
30	1/34	9/527	20/417	7/166	22/673	25/178	29/165
50	18/311	20/243	20/811	0/956	60/583	64/546	68/012
Griewank							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0/081	0/169	0/28	0/061	3/3	4/042	4/51
30	0/007	228/708	630/432	282/108	25/401	26/921	30/285
50	948/244	1053/5	1180/919	64/91	64/006	66/931	74/236
Rastrigin							
n	best	average	worst	variance	min_time	avg_time	max_time

10	2/985	13/432	35/818	8/304	2/977	3/093	3/652
30	41/87	366393/93	1840626	732424	22/784	23/651	24/491
50	468/99	2453097/4	3394777	1242255	62/361	65/773	70/785
Generalized Penalized							
n	best	average	worst	variance	min_time	avg_time	max_time
10	-3/065	2273442/9	15389408	4889766	5/798	6/182	7/254
30	-1/022	251187239	6/7E+08	2/62E+08	49/613	56/328	61/644
50	915027566	1/044E+09	1/25E+09	1/02E+08	144/241	154/673	173/673

- PSO Results with bounded velocity:

Rosenbrock							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0.371	2.897	7.328	2.22	3/534	3/724	3/925
30	46.455	109172085.95	27455607	11508433	30/413	31/808	34/178
50	423.671	300610421.6	48870024	20181143	81/514	85/15	88/959
Step							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0	1020.734	6461.495	2129.866	1/532	1/764	2/091
30	0	30204.419	69741.778	30895.7	10/902	11/979	13/843
50	108094.89	116438.34	123925	5939.35	29/053	31/056	32/958
Ackley							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0	3.738	19.317	7.481	3/071	3/155	3/31
30	16.644	19.144	20.699	1.339	22/673	25/178	29/165
50	19.094	20.136	20.863	0.75	60/583	64/546	68/012
Griewank							
n	best	average	worst	variance	min_time	avg_time	max_time
10	0.034	37.907	138.325	49.55	3/3	4/042	4/51
30	0.038	98.677	546.92	197.694	25/401	26/921	30/285
50	0.511	719.646	1123	476.1	64/006	66/931	74/236
Rastrigin							
n	best	average	worst	variance	min_time	avg_time	max_time
10	10.945	21074.108	210531.1	63152.34	2/977	3/093	3/652
30	130.234	485757.737	1663701.3	741979.4	22/784	23/651	24/491
50							
Generalized Penalized							
n	best	average	worst	variance	min_time	avg_time	max_time
10							
30							
50							

- PSO One-minute run Results:

Rosenbrock		
n	best	
10	0	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
30	79.348	[0.988, 0.977, 0.955, 0.912, 0.833, 0.694, 0.482, 0.229, 0.028, -0.627, 0.906, 1.047, 1.029, 1.016, 1.012, 1.013, 1.021, 1.04, 1.081, 1.04, 1.021, 1.012, 1.01, 1.013, 1.024, 1.046, 1.054, 1.192, 1.416, 2.004]
50	2658.411	[1.114, 1.152, 1.177, 1.194, 1.07, 0.895, 0.325, 0.17, 0.091, 0.067, -0.073, -0.042, 0.386, 0.979, 0.729, -0.009, -0.893, 1.079, 0.857, 0.564, 0.022, 1.372, 2.304, 1.548, 1.059, 0.665, 0.029, 0.104, -0.187, 0.369, 0.113, -0.439, 0.846, 1.016, 1.331, 1.091, 0.787, 0.721, 0.48, 0.234, 0.201, 0.301, 0.032, -0.035, 0.172, 0.048, -1.323, 0.772, -0.193, 0.234]
Step		
n	best	
10	0.0	[-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5]
30	61661.144	[-5.858, 5.602, -27.037, 88.178, -3.974, -55.965, 34.315, -43.495, 61.281, 5.569, 17.742, 5.192, 24.106, 20.063, -88.951, 6.362, 25.86, 3.915, -47.065, 45.11, -18.366, -108.129, -15.52, -82.158, -95.238, 44.802, -12.759, 91.51, -50.229, -50.748]
50	0.005	[-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.432, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.502, -0.5, -0.5, -0.5, -0.502, -0.5, -0.485, -0.5, -0.5, -0.5, -0.498, -0.5, -0.5, -0.5, -0.5, -0.496, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.502, -0.5, -0.501, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5]
Ackley		
n	best	
10	0	[0.0, -0.0, -0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
30	19.007	[1.0, -14.998, -38.995, -16.998, -21.997, -6.999, -11.999, 29.996, -7.999, 4.0, 7.999, 0.0, -10.999, -6.999, -2.0, -8.999, -7.999, -8.999, -12.998, 19.998, -1.0, 7.999, -18.998, -7.999, 7.999, -9.999, -25.997, -4.999, 13.998, -21.997]
50	20.659	[13.406, 24.386, 12.43, -30.978, -33.597, 8.828, 22.632, -10.782, 17.49, 2.285, -15.615, -28.179, 6.809, -8.869, -30.112, -20.477, -6.574, 8.922, 14.934, 5.427, 11.063, -5.604, 14.658, 11.491, -34.686, -22.39, -3.332, 6.457, 17.169, -16.106, -11.711, 21.702, -6.79, 13.995, -9.997, 14.195, -33.611, -1.921, -14.841, 17.232, 5.132, -5.156, -6.028, -34.355, 19.097, -7.047, 12.109, -4.295, -12.327, 26.478]
Griewank		
n	best	

10	0.221	[9.42, -0.0, -10.866, 6.271, -14.015, -0.0, 16.566, 8.85, -9.383, 0.0]
30	4.011	[3.14, 0.0, -5.433, 12.541, 0.0, -0.0, 0.0, 17.695, -0.0, -0.0, 10.361, -0.0, 123.787, -0.0, -0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0.0, -0.0, -0.0]
50	979.237	[-92.36, 254.498, 415.037, -343.41, -31.687, 28.422, -526.851, -658.36, 444.411, -406.794, 500.601, -214.744, 570.503, 66.734, 61.593, -235.204, -448.856, -273.234, -132.496, -206.161, -327.569, -226.706, -161.922, -5.701, 269.226, -155.727, 140.683, -497.707, -473.065, 199.034, -582.401, 74.843, 244.42, -175.244, 42.905, 22.555, -648.864, 327.402, 133.435, -271.31, -351.635, -192.33, 53.969, -322.471, -265.427, 86.231, -183.941, -29.449, -94.98, -346.925]
Rastrigin		
n	best	
10	15.919	[-0.0, -1.99, -0.0, -0.0, -2.985, 0.995, -0.995, -0.0, 0.995, 0.0]
30	147.252	[-2.985, 0.0, -0.995, 0.995, -2.985, 2.985, -0.995, 0.0, 0.995, 1.99, 0.0, -0.0, 0.0, -6.964, 1.99, -0.0, 0.995, -0.995, 0.0, 0.995, -0.995, -0.0, -2.985, -0.995, -0.0, -0.995, -1.99, -0.995, -5.97, -1.99]
50	1889.528	[2.983, 2.003, 0.991, -5.973, -0.01, 0.006, -3.96, 0.986, 1.079, 2.992, 37.44, -1.002, 1.994, -0.997, -0.947, 1.987, 2.984, 0.988, -1.017, 0.993, 0.001, -6.04, 4.998, -2.979, -4.091, 1.013, 5.974, -0.991, -0.986, 0.008, 1.995, -1.999, -1.984, -0.006, 3.977, 2.013, 0.005, 1.988, -0.987, 0.996, 4.966, -0.007, -0.993, 1.027, -1.99, -0.002, -6.965, 2.921, 0.984, -9.504]
Generalized Penalized		
n	best	
10	-2.449	[-6.881, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]
30	476743812.755	[2.916, -45.029, -32.395, 1.811, -49.305, 23.057, -27.949, -39.153, -42.559, -13.423, -1.431, -46.387, -39.859, 11.13, 7.566, -22.278, 19.796, 23.79, -52.133, 17.245, 3.086, -29.625, -18.617, -10.062, 15.482, -11.785, -7.635, 29.141, -19.703, -6.618]
50	807707467.747	[-18.073, -17.976, 1.63, 9.591, -17.199, 18.514, 19.88, -24.707, -22.914, 9.877, -11.192, -20.418, 8.075, 43.82, -41.172, 26.551, -32.989, 23.419, -29.898, -37.092, -0.446, -1.641, -16.665, 8.251, 6.151, 4.04, -49.188, 8.808, -34.595, 2.383, -7.025, -12.166, -9.567, 0.085, 38.619, -41.997, -15.842, -3.27, -4.7, -16.262,

		6.173, -12.139, -16.655, 21.764, -15.11, 20.135, -27.764, -53.662, 3.093, -38.806]

- PSO Algorithm analysis:

As algorithm description shows it has a lot of properties to be set and tune well and it need excremental researches to find the best possible configuration of algorithm.

Its good practice to set a little higher exploration rate(c1) for problems with unknown search space But in opposite higher exploitation rate(c2) is more suitable for known problems search area.

Types of inertial weight update methods have their own spatial characters and it’s hard to compare their performance, I think the choice could be dependent to search space area (Hight local optima positions or low).

Overall, I think this approach advantages would be good searching process (act good to find global optima) and simple implementation.
But its disadvantage would be lot of properties to config and manage.