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. |  |
| **E** | Rosenbrock | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |
| Step | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |
| **M** | Ackley | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |
| Griewank | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |
| **H** | Rastrigin | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |
| Generalized Penalized | C:\Users\somayeh\Desktop\Untitled.png |  |  | 0 |

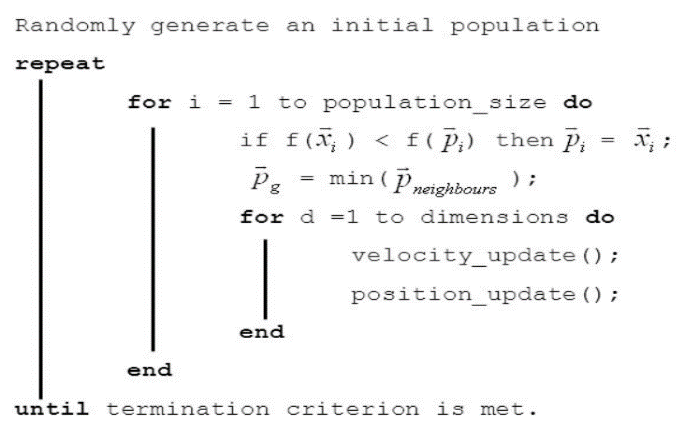
**Description:**

denotes the solution space dimension, denotes a subset of , and the global optimal solution and the global optimal value of classical benchmark functions are given in column 5 and column 6, respectively. Ten independent experiments must be completed for each optimization function considering .

* 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.



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.

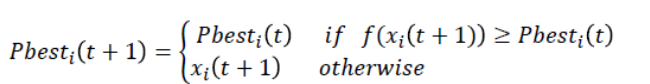
* + **Position & velocity update:**

**

*d* is the dimension, *c1* and *c2* are positive constants, *rand1* and *rand2* are random numbers, and *w* is the inertia weight.

usually *c1+ c2 = 4*. No good reason other than empiricism.

* + **Pbest & Gbest update:**





* + **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

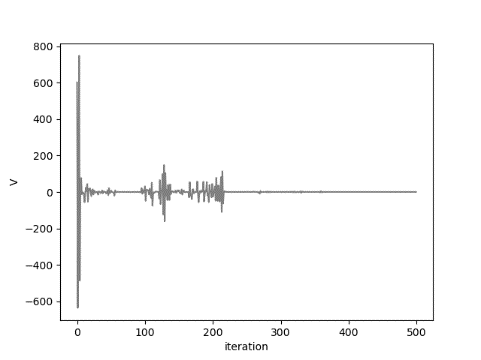
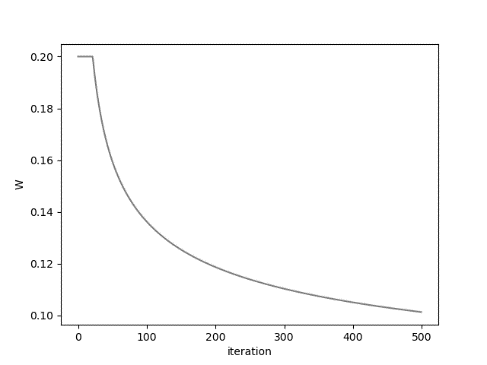
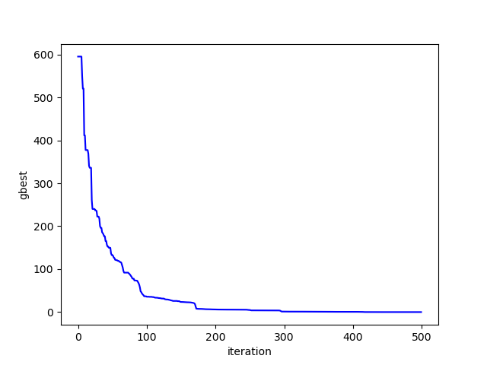
SWARM\_SIZE = 80

wMax = 0.9

wMin = 0.2

inertia mode = logarithmic

* + **Griewank**



Gbest /iteration inertia weight /iteration velocity/iteration

* + **Griewank**
* PSO Results:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 One-minute run Results:

|  |  |
| --- | --- |
| Rosenbrock |  |
| n | best |
| 10 | 0 |
| 30 | 16/131 |
| 50 | 468914323 |
|  |  |
| Step |  |
| n | best |
| 10 | 8683/25 |
| 30 | 67776/246 |
| 50 | 115852/74 |
|  |  |
| Ackley |  |
| n | best |
| 10 | 0 |
| 30 | 20/605 |
| 50 | 18/72 |
|  |  |
| Griewank |  |
| n | best |
| 10 | 0/052 |
| 30 | 0/248 |
| 50 | 1115/439 |
|  |  |
| Rastrigin |  |
| n | best |
| 10 | 13/929 |
| 30 | 1587320/7 |
| 50 | 3068987/9 |
|  |  |
| Generalized Penalized | |
| n | best |
| 10 | -3/065 |
| 30 | 524236888 |
| 50 | 49688/731 |

* 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.