Technical Report - Project B - Optimization Techniques

Mahan RajaeiGolsefidi

School of Informatics, Aristotle University of Thessaloniki

Spring 2022

Question 1.

Particle swarm optimization (PSO) is a nature-inspired algorithms. PSO algorithms search for an optimal solution in the solution space. It is different from other optimization algorithms in such a way that only the objective function is required, and it does not rely on the gradient or any differential form of the objective. It also has very few hyperparameters. While using PSO, we can never prove the real **global optimal** solution can be found. However, we often find that the solution found by PSO is quite close to the global optimal.

The basic PSO can be describes as follows:

f: $\mathbb{R}^n \rightarrow \mathbb{R}$ ➔ the objective function which must be maximized.

Let S be the number of particles in the swarm.

Each particle has a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$.

Let $\mathbf{x}_i^*$ be the best known position of particle i.

Let **g** be the best known position of the entire swarm.

Initially, each particle takes a random position $\mathbf{x}_i^0$ within the search-space. Also, initial velocities can be set to zero ($\mathbf{v}_i^0 = \mathbf{0} \in \mathbb{R}^n$).

Let K be the maximum number of iterations.

At each step (k^{th} step), the algorithm updates the position of the particles:

$$\boldsymbol{x_i^k = x_i^{k-1} + v_i^k}$$

where,

$$\boldsymbol{v_i^k = \theta^k v_i^{k-1} + c_1 r_{1i}^k (x_i^* - x_i^{k-1}) + c_2 r_{2i}^k (g - x_i^{k-1})}$$

$$\boldsymbol{\theta^k = \theta^{max} - \left(\frac{\theta^{max} - \theta^{min}}{K}\right).k}$$

$\boldsymbol{\theta}$ is the inertia at each step which is initially set to θ^{max} and descends to θ^{min} as the algorithm approaches its maximum number of iterations. If $\boldsymbol{\theta = 1}$, the particle's motion is entirely influenced by the previous motion, so the particle may keep going in the same direction. If $0 \leq \boldsymbol{\theta} < \mathbf{1}$, such influence is reduced, which means that a particle instead goes to other regions in the search domain.

the idea behind the term $\boldsymbol{(x_i^* - x_i^{k-1})}$ is that as the particle gets more distant from the $\boldsymbol{x_i^*}$ (local/personal best) position, the difference $\boldsymbol{(x_i^* - x_i^{k-1})}$ must increase; hence, this term increases, attracting the particle to its best

own position. The parameter $\boldsymbol{c_1}$ is a positive constant, and it is an individual-cognition parameter. It weighs the importance of the particle's own previous experiences.

$\boldsymbol{r_1}$ is a random value parameter with range [0, 1]. This random parameter plays an essential role in avoiding premature convergences, increasing the most likely global optima.

The term $\left(\boldsymbol{g} - \boldsymbol{x_i^{k-1}}\right)$ works as an attraction for the particles towards the best point (global best). Likewise, $\boldsymbol{c_2}$ is also a social learning parameter, and it weighs the importance of the global learning of the swarm. And $\boldsymbol{r_2}$ plays the same role as $\boldsymbol{r_1}$.

In this project, for each particle, in each iteration, $\boldsymbol{r_{1i}^k}$ and $\boldsymbol{r_{2i}^k}$ are uniformly distributed random numbers in (0, 1), and $\boldsymbol{c_1} = \boldsymbol{c_2} = \frac{\mathbf{1}}{\mathbf{2}}$.

Additionally, since we are dealing with a minimization problem, we use a substitute function to maximize:

$$\boldsymbol{F(x)} = \frac{\mathbf{1}}{\mathbf{0.1} + \boldsymbol{f(x)}}$$

Pseudo code:

for each particle $i = 1, ..., S$ **do**
  $\mathbf{x}_i \sim U$(lower_boundary, upper_boundary)
  $\mathbf{v}_i \leftarrow \mathbf{0}$ (vector of size n)
  Initialize the particle's best known position to its initial position: $\mathbf{x_i}^* \leftarrow \mathbf{x}_i$
  if $F(\mathbf{x_i}^*) > F(\mathbf{g})$ **then**
    update the swarm's global best position: $\mathbf{g} \leftarrow \mathbf{x_i}^*$
k = 0
while a termination criterion is not met (finding solution or reaching maximum iteration) **do**:
  k = k + 1
  for each particle $i = 1, ..., S$ **do**
    for each dimension $d = 1, ..., n$ **do**
      Pick random numbers: $r_1, r_2 \sim U(0,1)$
      Update the particle's velocity: $\boldsymbol{v_{di}^k} \leftarrow \theta^k \boldsymbol{v_{di}^{k-1}} + \boldsymbol{c_1 r_{1i}^k}\left(\boldsymbol{x_{di}^*} - \boldsymbol{x_{di}^{k-1}}\right) + \boldsymbol{c_2 r_{2i}^k}\left(\boldsymbol{g_d} - \boldsymbol{x_{di}^{k-1}}\right)$
    Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
    if $F(\mathbf{x}_i) > F(\mathbf{x_i}^*)$ **then**
      Update the particle's local best position: $\mathbf{x_i}^* \leftarrow \mathbf{x}_i$
      if $F(\mathbf{x_i}^*) > F(\mathbf{g})$ **then**
        Update the swarm's global best position: $\mathbf{g} \leftarrow \mathbf{x_i}^*$

Minimization objective functions:

Function #1: Sphere Function

$$f = \sum_{i=1}^{n} x_i^2$$

Where, n = 2, 10, 20

The python implementation of the PSO algorithm is attached to this report. The code is structurally simple and straightforward and is extensively commented to provide maximum information.

Two symmetrical boundary cases are examined: [-10, 10] and [-500, 500]

Figures 1 through 12 illustrate the results.

```
In [1]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 2

Symetrical Boundary: [-10, 10]

Number of Particles: 30

Maximum Number of Steps: 5000


################## Results ##################

Number of Steps: 416

Optimal X vector:
[-7.168251826791122e-05, -0.0009098800003527275]

Optimal Function Value: 8.330199984671087e-07

Difference between Found Optimal Value and True Optimal Value: 8.330199984671087e-07

Euclidean Distance from True Solution: 0.0009126992924655462
```

Figure 1. Result log. Function #1 ; Dimension = 2 ; Narrow Boundary

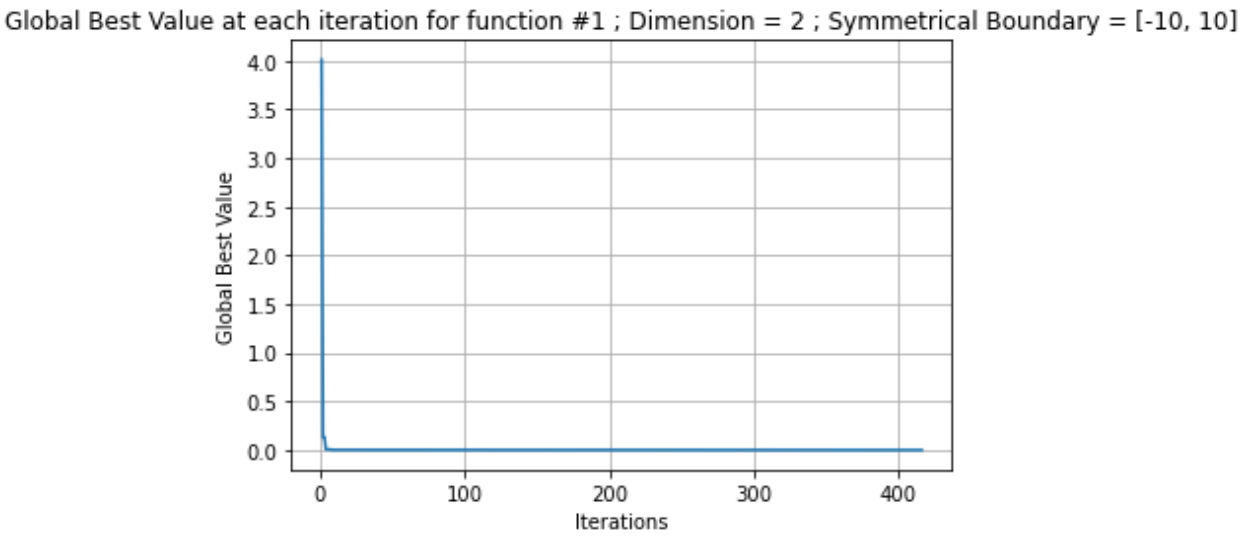

Figure 2. Global Best Diagram

```
In [2]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 2

Symetrical Boundary: [-500, 500]

Number of Particles: 30

Maximum Number of Steps: 5000


################## Results ##################

Number of Steps: 684

Optimal X vector:
[1.5022521578363784e-05, 0.00011861492608088078]

Optimal Function Value: 1.4295176843745217e-08

Difference between Found Optimal Value and True Optimal Value: 1.4295176843745217e-08

Euclidean Distance from True Solution: 0.00011956243910085315
```

Figure 3. Result log. Function #1 ; Dimension = 2 ; Wide Boundary

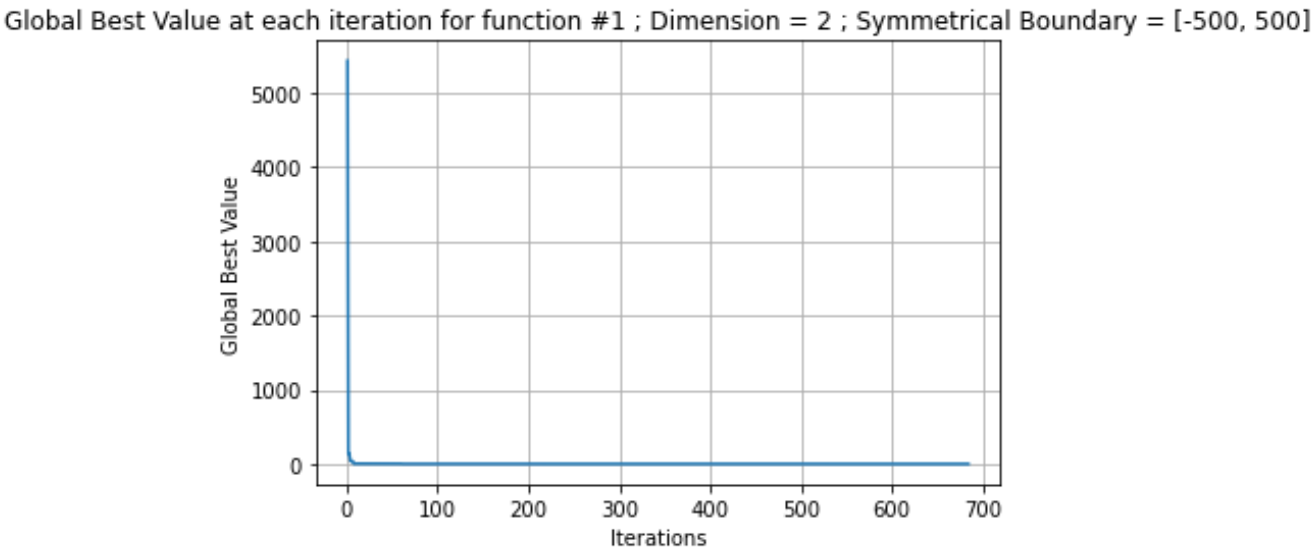

Figure 4. Global Best Diagram

```
In [3]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 10

Symetrical Boundary: [-10, 10]

Number of Particles: 41

Maximum Number of Steps: 5000


################# Results #################

Number of Steps: 1110

Optimal X vector:
[9.267092680345992e-05, -0.00044425814297775583, 0.0005006864355409913, 2.0510848534526015e-05, 0.00030270430026660953, 0.0005393775833085694, -4.171058518334355e-06, 6.619660820637299e-05, -0.0002350866916786968, 0.00022258275790263977]

Optimal Function Value: 9.488270960818674e-07

Difference between Found Optimal Value and True Optimal Value: 9.488270960818674e-07

Euclidean Distance from True Solution: 0.0009740775616355544
```

Figure 5. Result log. Function #1 ; Dimension = 10 ; Narrow Boundary

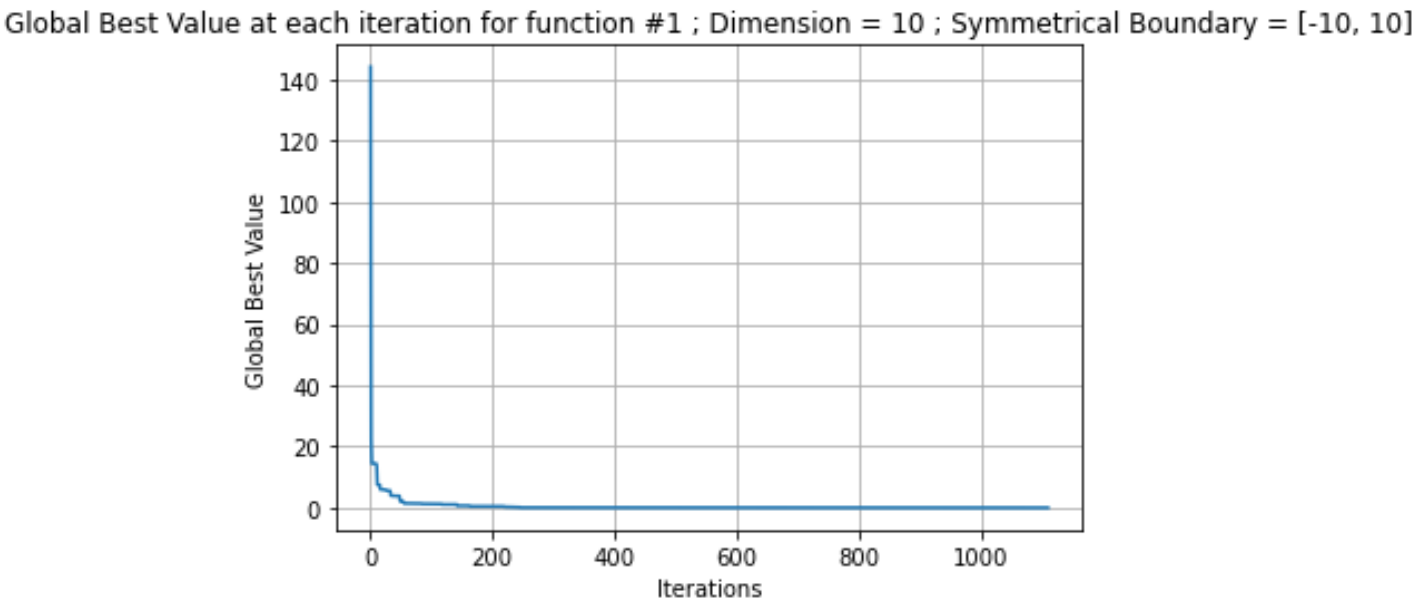

Figure 6. Global Best Diagram

```
In [4]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 10

Symetrical Boundary: [-500, 500]

Number of Particles: 41

Maximum Number of Steps: 5000


################## Results ##################

Number of Steps: 1574

Optimal X vector:
[0.00029482486238426714, 0.00023038068352949445, -0.00050311155239791O1, 9.667383319111236e-05, 0.00016080308537708236, 0.00051789640809190 11, 0.00015648244998824631, -0.00032085498190307824, 0.00032183945303629054, 6.353109602785182e-05]

Optimal Function Value: 9.315898550443468e-07

Difference between Found Optimal Value and True Optimal Value: 9.315898550443468e-07

Euclidean Distance from True Solution: 0.0009651890255511336
```

Figure 7. Result log. Function #1 ; Dimension = 10 ; Wide Boundary

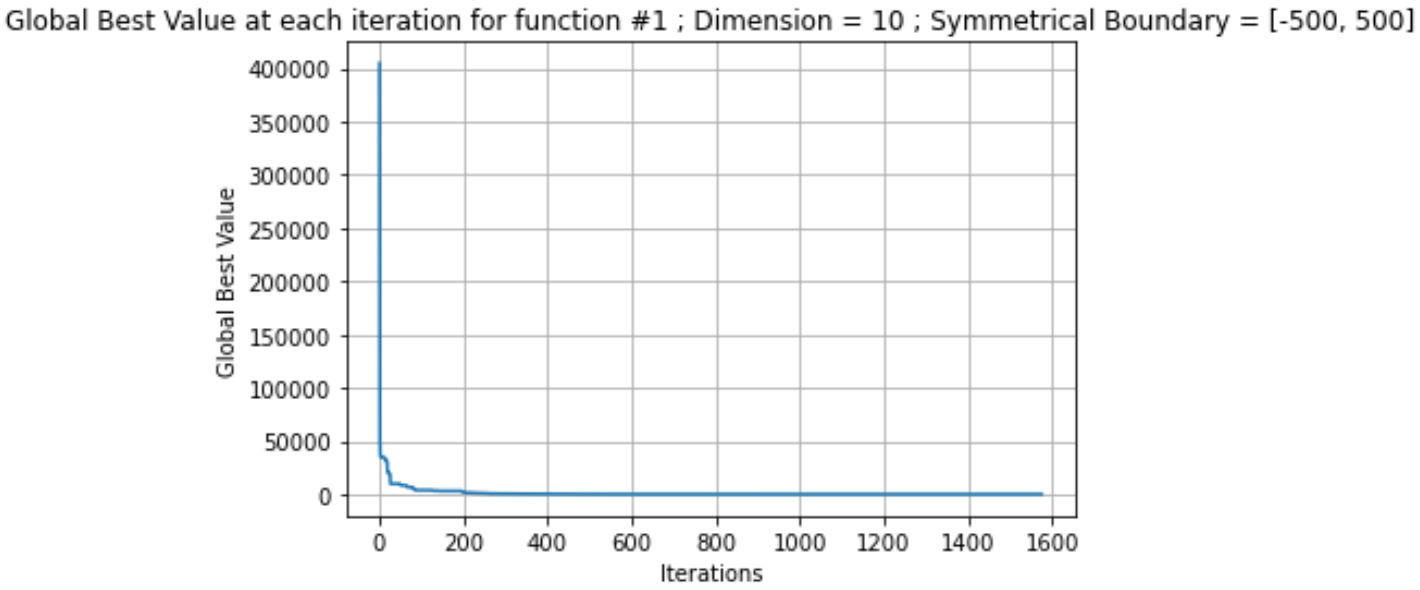

Figure 8. Global Best Diagram

```
In [5]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 20

Symetrical Boundary: [-10, 10]

Number of Particles: 81

Maximum Number of Steps: 5000


################## Results ##################

Number of Steps: 1681

Optimal X vector:
[0.00013425777720770921, -0.00023715623877493098, -0.00054359845225550371, -0.00024193837553796122, 0.00032392255348215023, -0.00015540865227354106, 1.528048788033558e-05, -0.00013113320192052345, -1.7197367917949584e-05,
2.2555799859213053e-05, 0.00010546251183871719, -0.00010660066140084733, -4.7714815169914e-05, 0.00016315741584579897, 0.00020496066744380055, -0.00035771096309161385, 0.00034373625764962405, -4.307174700736558e-05, -0.00021879365315908O1,
-0.00018216238550246591

Optimal Function Value: 9.980259634143O5e-07

Difference between Found Optimal Value and True Optimal Value: 9.98025963414305e-07

Euclidean Distance from True Solution: 0.0009990124941232242
```

Figure 9. Result log. Function #1 ; Dimension = 20 ; Narrow Boundary

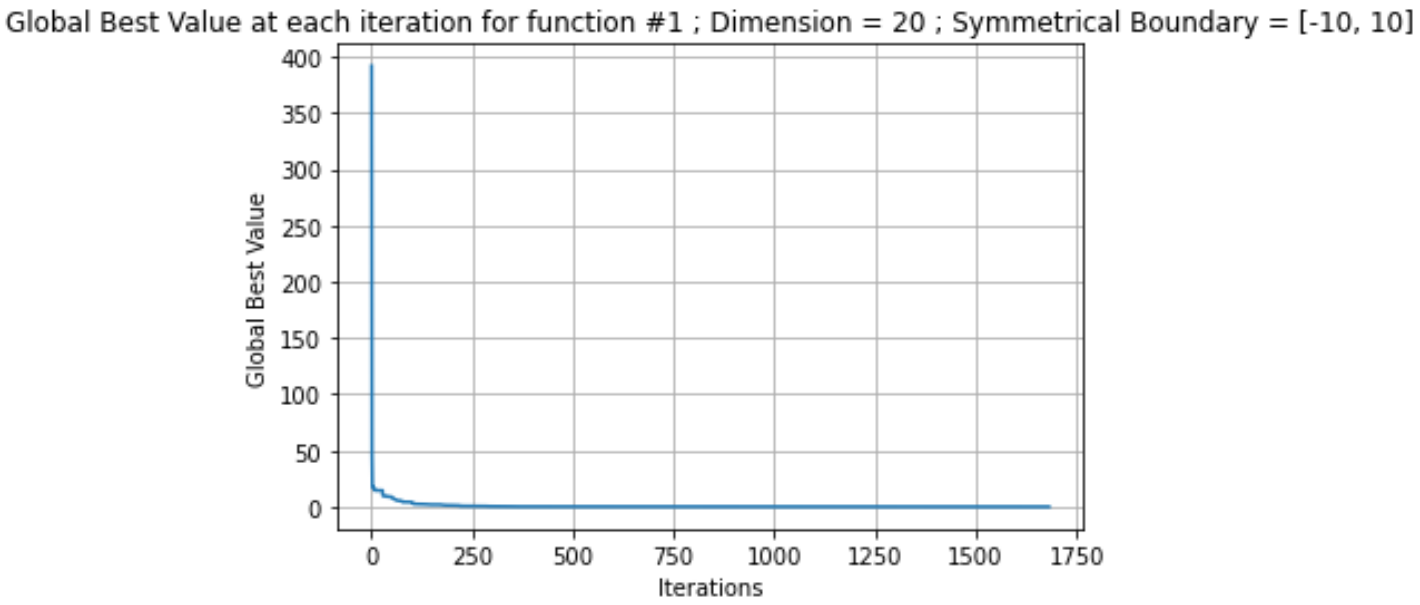

Figure 10. Global Best Diagram

```
In [7]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 1

Dimension: 20

Symetrical Boundary: [-500, 500]

Number of Particles: 81

Maximum Number of Steps: 10000


################## Results ##################

Number of Steps: 2869

Optimal X vector:
[-5.742404515120313e-05, 0.0001251844135978851, 9.021716706615715e-05, 7.065456243887056e-05, 0.00015361322239777102, -3.280863931081875e-06, -0.0001404943172712267 7, -0.0003197497225819887 4, 0.0003427254758324722 7, -6.534572997157306e-05,
0.00014768066660296756, 0.00010717782867351595, -0.00018343532832580754, 9.296080538987456e-05, 0.0002369927570520376, 0.0002064204409270021 8, 0.0004123396321180948, -1.0896684503701813e-05, -0.0001877647380547082, 0.0005078489719858798]

Optimal Function Value: 9.370877511908339e-07

Difference between Found Optimal Value and True Optimal Value: 9.370877511908339e-07

Euclidean Distance from True Solution: 0.000968032928774034
```

Figure 11. Result log. Function #1 ; Dimension = 20 ; Wide Boundary

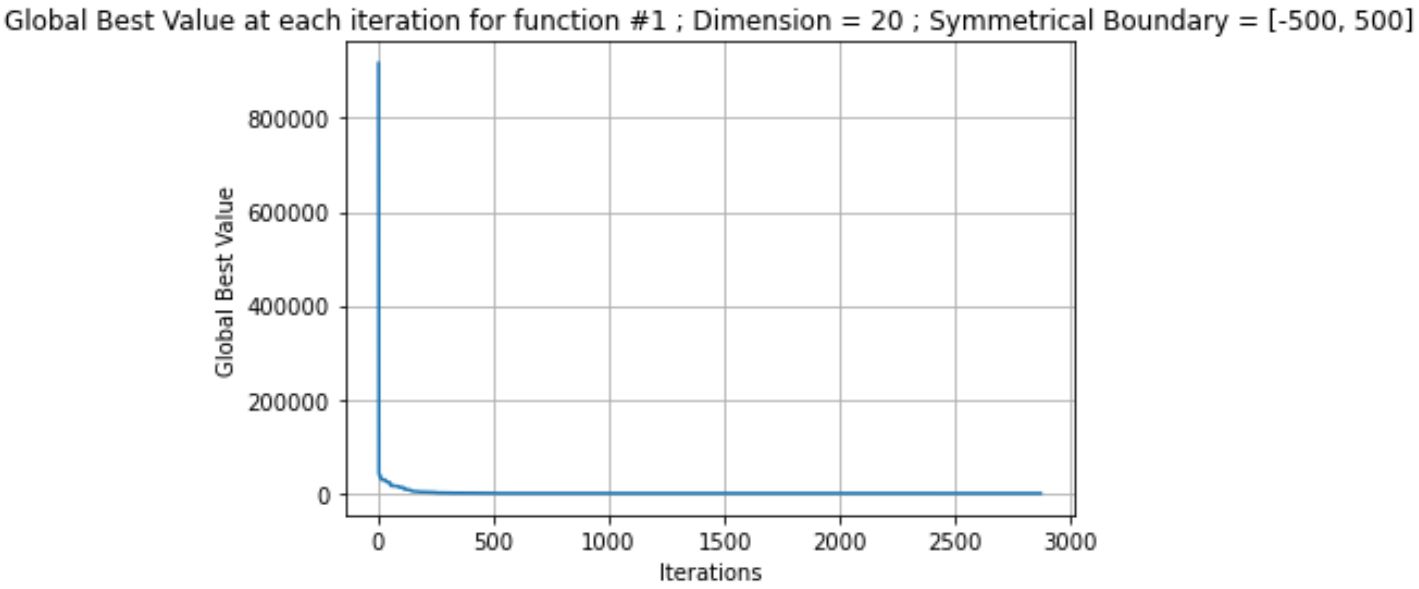

Figure 12. Global Best Diagram

Function #2: Booth Function

$$f(x,y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

One symmetrical boundary is examined: [-10, 10].

Figures 13 and 14 illustrate the results.

```
In [8]: runfile('/home/ryan/Desktop/0_0_Project_B/Codes/Question1.py', wdir='/home/ryan/Desktop/0_0_Project_B/Codes')
###### Function number 2

Dimension: 2

Symetrical Boundary: [-10, 10]

Number of Particles: 30

Maximum Number of Steps: 10000


################## Results ##################

Number of Steps: 605

Optimal X vector:
[0.9996267534038423, 3.0004893803611346]

Optimal Function Value: 4.3275436487833104e-07

Difference between Found Optimal Value and True Optimal Value: 4.3275436487833104e-07

Euclidean Distance from True Solution: 0.0006154723059630479
```

Figure 13. Result log. Function #2 ; Dimension = 2

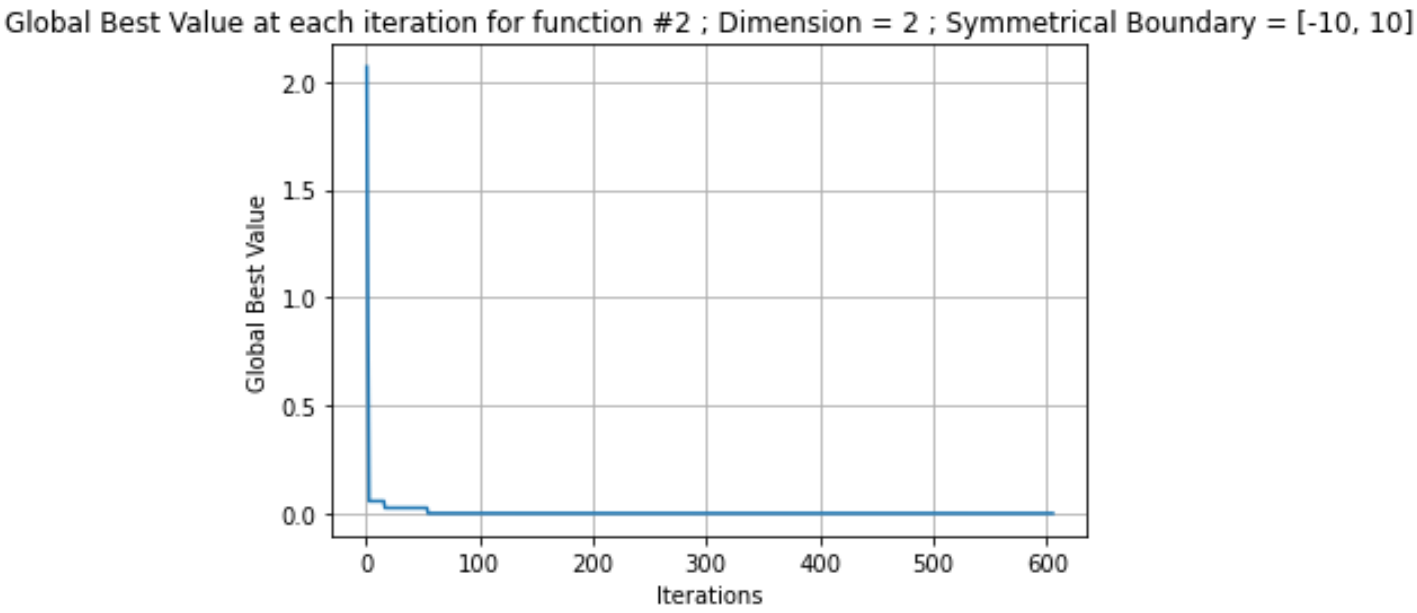

Figure 14. Global Best Diagram