Particle Swarm Optimization: Intelligent Parameter Tuning

Armand Maree
Department of Computer Science
University of Pretoria

***Abstract*—Particle Swarm Optimization (PSO) is a optimization algorithm that models the flocking behaviour of certain species of animals in order to solve stochastic real-value problems. These algorithms use control parameters to make slight adjustments to how the swarm behaves. In this report the author will be discussing the results of using a brute force approach to find these control parameters.**

I. Introduction

A Particle Swarm Optimisor (PSO) is, as described by Lazincia [1], an optimization algorithm that attempts to emulate the flocking/schooling behaviour of birds or fish when searching for food. This optimization can occur in a number of dimensions, each spanning across $\mathbb{R}$ [2]. Each particle in the swarm has two primary components of influence: cognative and social. A third component, called the inertia weight, is used to regulate the maximum velocity that a particle can move at in a certain direction [3]. In this report, the author will explore the feasibility of finding the optimum value for each of these components by using a brute force search in a subset.

II. Background

As discussed in section I, the two components that influence a particle's behaviour is cognative and social. The cognative component refers to the influence that a particle experiences due to the best position it has found so far [4]. A more animalistic analogy of this would be that if you feed a bird it will be more inclined to visit your house in the future to get more food. The social component refers to how a particle is influenced by the best position found by other particles in the neighbourhood [4]. The analogy here would be that if you feed one bird the other birds in the flock might be more inclined to also visit your house to get food. There is a trade-off between these two components. The cognative component favours exploration of the search space, while the social component favours exploitation of the global best position.

The last parameter is the inertia weight developed by Shi and Eberhart (1998). The role of this parameter is to balance global search and local search. [4] and thus controls the velocity a particle can move at. The inertia weight was originally proposed in order to avoid the velocity clamping mechanism that was used before.

In order to test the feasibility of the brute force approach four objective functions were used.

- Spherical

$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 \tag{1}$$

- Ackley

$$f(\mathbf{x}) = -20e^{-0.2\sqrt[2]{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} + 20 + e \tag{2}$$

- Michalewicz

$$f(\mathbf{x}) = -\sum_{i=1}^{d}[\sin(x_i)\sin^{2*10}(\frac{ix_i^2}{\pi})] \tag{3}$$

- Katsuura

$$f(\mathbf{x}) = \frac{10}{d^2\prod_{i=1}^{d}}(1 + i\sum_{j=1}^{32}\frac{\left|2^j x_i - round(2^j x_i)\right|}{2^j})^{\frac{10}{d^{1.2}}} - \frac{10}{d^2} \tag{4}$$

The topology that was used is the star topology, as prescribed by the project specifications. This model states that all the particles in the swarm are connected to all other particles in the swarm. This means that there is only one neighbourbood and all particles belong to it. This approach means that the global best position (gbest) will be used as the current optimum.

The brute force approach used in this report refers to the systematic exhaustive evaluation of a parameter that falls in a finite set of numbers. Since the range that these three parameters usually fall in is known and small, it might prove to be a feasible option to find these parameters via this approach.

III. Implementation

A swarm of 20 particles was used to solve all the objective functions in a 20-dimensional space. The restrictions on w, c_1 and c_2 was specified as follows:

- $w \in [-1.1, 1.1]$
- $c_1 + c_2 \in (0.0, 5.0]$
- $c_1 = c_2$

Each of these values had a step size of 0.1.

The velocity, $\mathbf{v}_i$, of a particle, par_i, will be used to update the position, $\mathbf{x}_i$, of par_i, as follows:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{5}$$

where,

$$\begin{aligned}\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \\ c_2\mathbf{r}_2 \otimes (\mathbf{g}_i(t) - \mathbf{x}_i(t))\end{aligned} \tag{6}$$

where,

$\mathbf{p}_i$ refers to the personal best position of par_i,
and,
$\mathbf{g}_i$ refers to the global best position of the entire swarm.

$\otimes$ refers to componentwise multiplication of vectors.
w used in equation 6 is the inertia weight as discussed in section I.
c_1 and c_2 used in equation 6 represents the cognative and social components respectively as discussed in section I.

Since PSO is a stochastic algorithm a single run can have abnormally good or bad results. Each set of possible combinations between w, c_1 and c_2 has been tested 30 times and the average result of all 30 runs has been taken as the optimal result for the specific configuration. The best configuration's result, along with the corresponding values for w, c_1 and c_2 is stored each time a new optimum is found.

A simple pseudo code version of the entire algorithm is shown in algorithm 1 and 2:

```
initializeSwarm()
while not (stopping condition) do
    foreach particle in swarm do
        var objectiveFunctionResult =
         particle.getObjectiveFunctionResult() if
         objectiveFunctionResult < particle.personalBest
         then
            /* Save current position as personal best */
        end
        if objectiveFunctionResult <
         particle.neighbourhoodBest then
            /* Save current position as neighbourhood
              best */
        end
    end
    foreach particle in swarm do
        particle.updateVelocity() particle.updatePosition()
    end
end
return particle[0].neighbourhoodBest
```

Algorithm 1: Main PSO Algorithm

```
var bestSolution = infinity for w <- -1.1 to 1.1 do
    for c1 <- 0.1 to 2.5 do
        c2 = c1
        for currentRun <- 1 to 30 do
            /* call "solvePSO" function which is
              algorithm 1 */
            var bestResult +=
              objectiveFunction(solvePSO())
        end
        if bestResult/30 < bestSolution["result"] then
            bestSolution["result"] = bestResult
            bestSolution["w"] = w
            bestSolution["c1"] = c1
            bestSolution["c2"] = c2
        end
    end
end
```

Algorithm 2: PSO Parameter Finding

IV. Research Results

The algorithm managed to obtain values for all the parameters to all the objective functions. The results were as follows:

- Spherical
  - $Optimum \approx 0.0$
  - Variance across 30 runs ≈ 0.0
  - $w = 0.6$
  - $c1 = 0.6$
  - $c2 = 0.6$
  - $x_0 \approx 0.0$
  - $x_1 \approx 0.0$
  - $x_2 \approx 0.0$
  - $x_3 \approx 0.0$
  - $x_4 \approx 0.0$
  - $x_5 \approx 0.0$
  - $x_6 \approx 0.0$
  - $x_7 \approx 0.0$
  - $x_8 \approx 0.0$
  - $x_9 \approx 0.0$
  - $x_{10} \approx 0.0$
  - $x_{11} \approx 0.0$
  - $x_{12} \approx 0.0$
  - $x_{13} \approx 0.0$
  - $x_{14} \approx 0.0$
  - $x_{15} \approx 0.0$
  - $x_{16} \approx 0.0$
  - $x_{17} \approx 0.0$
  - $x_{18} \approx 0.0$
  - $x_{19} \approx 0.0$
- Ackley
  - $Optimum \approx 0.0$
  - Variance across 30 runs ≈ 0.0
  - $w = 0.6$
  - $c1 = 0.6$
  - $c2 = 0.6$
  - $x_0 \approx 0.0$
  - $x_1 \approx 0.0$
  - $x_2 \approx 0.0$
  - $x_3 \approx 0.0$

- $x_4 \approx 0.0$
- $x_5 \approx 0.0$
- $x_6 \approx 0.0$
- $x_7 \approx 0.0$
- $x_8 \approx 0.0$
- $x_9 \approx 0.0$
- $x_{10} \approx 0.0$
- $x_{11} \approx 0.0$
- $x_{12} \approx 0.0$
- $x_{13} \approx 0.0$
- $x_{14} \approx 0.0$
- $x_{15} \approx 0.0$
- $x_{16} \approx 0.0$
- $x_{17} \approx 0.0$
- $x_{18} \approx 0.0$
- $x_{19} \approx 0.0$

- Michalewicz
  - $Optimum \approx -19.786$
  - Variance across 30 runs ≈ 0.0
  - $w = 0.6$
  - $c1 = 0.6$
  - $c2 = 0.6$
  - $x_0 \approx 623.61$
  - $x_1 \approx 1.57$
  - $x_2 \approx 1.28$
  - $x_3 \approx 1.92$
  - $x_4 \approx 1.72$
  - $x_5 \approx 1.57$
  - $x_6 \approx 7.83$
  - $x_7 \approx 1.76$
  - $x_8 \approx 1.28$
  - $x_9 \approx 1.57$
  - $x_{10} \approx 1.5$
  - $x_{11} \approx 1.43$
  - $x_{12} \approx 1.63$
  - $x_{13} \approx 1.57$
  - $x_{14} \approx 1.52$
  - $x_{15} \approx 1.67$
  - $x_{16} \approx 1.62$
  - $x_{17} \approx 1.39$
  - $x_{18} \approx 1.53$
  - $x_{19} \approx 1.49$
- Katsuura
  - $Optimum \approx 0.04$
  - $w = 0.6$
  - $c1 = 0.6$
  - $c2 = 0.6$
  - $x_0 \approx -1.0 \times 10^4$
  - $x_1 \approx -1.5 \times 10^6$
  - $x_2 \approx 2.1 \times 10^6$
  - $x_3 \approx -1.8 \times 10^4$
  - $x_4 \approx 1.6 \times 10^6$
  - $x_5 \approx -2.1 \times 10^6$
  - $x_6 \approx 3.2 \times 10^5$
  - $x_7 \approx 1.8 \times 10^6$
  - $x_8 \approx 1.0 \times 10^7$
  - $x_9 \approx 1.3 \times 10^6$
  - $x_{10} \approx 2.6 \times 10^6$
  - $x_{11} \approx 5.2 \times 10^5$
  - $x_{12} \approx -1.8 \times 10^5$
  - $x_{13} \approx -3.3 \times 10^4$
  - $x_{14} \approx -3.0 \times 10^6$
  - $x_{15} \approx 1.9 \times 10^4$
  - $x_{16} \approx 1.7 \times 10^4$
  - $x_{17} \approx 5.7 \times 10^5$
  - $x_{18} \approx 4.8 \times 10^5$
  - $x_{19} \approx 5.2 \times 10^6$

V. Conclusion

As expected, the results in section IV indicate, if a small range for each of the three parameters is known, it is possible to find the optimum value for an objective function. This can even be done is a fairly short amount of time if the objective function is simple, as is the case with equation 1, 2 and 3, but as the complexity (or rather computational time) grows, like with equation 4, this process can become quite lengthy.

Thus, if an equation does not contain a large amount of computational demanding steps, this brute force approach might be a feasible option since it is simple and quick to implement.

References

[1] Aleksandar Lazinica. *Particle swarm optimization*. InTech Kirchengasse, 2009.

[2] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104–4108. IEEE, 1997.

[3] Russ C Eberhart and Yuhui Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 84–88. IEEE, 2000.

[4] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998.

Appendix

Here follows some screenshots of the results outlined in section IV.

```
Swarm of size 20 created.
Number of dimensions: 20.
Using function 0.
Progress: 100%
Best solution found was:
        w:  0.600000
        c1:  0.600000
        c2:  0.600000
        Average:  0.000000
        Variance: 0.000000
With values:
        x0=-0.000000
        x1=-0.000000
        x2=-0.000000
        x3=-0.000000
        x4=0.000000
        x5=0.000000
        x6=0.000000
        x7=-0.000000
        x8=0.000000
        x9=0.000000
        x10=0.000000
        x11=0.000000
        x12=0.000000
        x13=0.000000
        x14=0.000000
        x15=-0.000000
        x16=-0.000000
        x17=0.000000
        x18=-0.000000
        x19=0.000000
```

Figure 1. Spherical Objective Function Results

```
Swarm of size 20 created.
Number of dimensions: 20.
Using function 1.
Progress: 100%
Best solution found was:
        w:  0.600000
        c1:  0.600000
        c2:  0.600000
        Average:  0.000000
        Variance: 0.000000
With values:
        x0=-0.000000
        x1=-0.000000
        x2=0.000000
        x3=-0.000000
        x4=0.000000
        x5=0.000000
        x6=-0.000000
        x7=0.000000
        x8=-0.000000
        x9=0.000000
        x10=-0.000000
        x11=-0.000000
        x12=0.000000
        x13=-0.000000
        x14=-0.000000
        x15=-0.000000
        x16=-0.000000
        x17=0.000000
        x18=0.000000
        x19=0.000000
```

Figure 2. Ackley Objective Function Results

```
Swarm of size 20 created.
Number of dimensions: 20.
Using function 2.
Progress: 100%
Best solution found was:
        w:  0.600000
        c1:  0.600000
        c2:  0.600000
        Average:  -19.785718
        Variance: 0.000000
With values:
        x0=623.608120
        x1=1.570796
        x2=1.284992
        x3=1.923058
        x4=1.720470
        x5=1.570796
        x6=7.831510
        x7=1.756087
        x8=1.282824
        x9=1.570796
        x10=1.497729
        x11=1.433992
        x12=1.630076
        x13=1.570796
        x14=1.517546
        x15=1.666065
        x16=1.616329
        x17=1.385349
        x18=1.528907
        x19=1.490199
```

Figure 3. Michalewicz Objective Function Results

```
Swarm of size 20 created.
Number of dimensions: 20.
Using function 3.
Progress: 100%
Best solution found was:
        w:  0.600000
        c1:  0.600000
        c2:  0.600000
        Average:  0.044864
        Variance: 0.000000
With values:
        x0=-10197.078117
        x1=-1450801.012048
        x2=2093022.436935
        x3=-17785.405925
        x4=1577409.499544
        x5=-2126971.500000
        x6=322336.032864
        x7=1812409.000200
        x8=10299924.500109
        x9=1398294.999034
        x10=2555499.500790
        x11=520024.500748
        x12=-178405.999528
        x13=-33450.003639
        x14=-2967462.999554
        x15=18790.000020
        x16=16821.006230
        x17=568735.499276
        x18=475856.501122
        x19=5214075.000000
```

Figure 4. Katsuura Objective Function Results