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 paper 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 paper, 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 [**?**], 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 convinced 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 parametr is to balance the global search and local search [4] and thus controls the velocity a particle can move at.

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

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

- 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 a global best position (gbest) will be used as the current optimum.

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,

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

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

V. Conclusion

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.