Cooperative PSO with Spatially Meaningful Neighbors

Peter Wilson

Supervised by Beatrice M. Ombuki-Berman

Submitted in partial fulfillment

of the requirements for COSC 4F90

Department of Computer Science

Brock University

St. Catharines, Ontario

Abstract

Contents

List of Tables

List of Figures

1 Introduction

2 Background

2.1 Particle Swarm Optimization

2.1.1 Overview

Particle Swarm Optimization (PSO) is a iterative population-based computational intelligence algorithm to find optimal solutions for non-linear continuous problems. Computational intelligence is a domain of study in which the patterns and behaviours of intelligent agents (such as humans or animals) are used to help solve other types of problems. Particle Swarm Optimization specifically draws from the flight patterns of birds in their search of food as well as the schooling pattern of fish. In both cases, the animals tend to follow each other from a distance which allows each to get a different vantage point. When one of the fish/birds find the best source of food (or the best place to land) they all tend to converge and head toward that point.

2.1.2 Particle

PSO builds off this metaphor by representing the birds or fish as Particles. A Particle has a position, velocity and personal best value. It keeps track of all these values and and updates them with each iteration. the position of the particle maps logically to the location of where the bird is on a map. The x,y,z,... values of the position also correlate to the input values in the problem which we are trying to optimize. The more input values (variables) the problem contains, the higher dimensions are needed to represent the problem. Velocity represents the direction and speed of the Particle. It determines how you update the position after each iteration. The velocity, too is updated at the end of each iteration and is influenced by the current velocity, the personal best value (or the best combination of values found previously by the particle which is maintained by the particle) and the global best (the best value found by all the particles in the swarm). The velocity and position are updated with the following functions:

position

velocity

2.1.3 Swarm

The Swarm is the heart that makes PSO work. It is the combination of a number of particles working in unison to solve the same problem. The Swarm needs to maintain and keep track of the individual particles but it also needs to store a Global Best value. The global best is the position of the best solution found by any of the particles within the swarm. Each particle will in turn use that value to help influence the change in velocity for each iteration. This transfer of knowledge amongst the particles helps the swarm converge on the optimal solution and

2.1.4 Fitness

1.

Table 1: GP Algorithm Pseudo Code

$pop \leftarrow randomPopulation(popSize)$;
for $gen \leftarrow 0, maxGen$ **do**
    $evaluateFitness(pop)$;
    $parents \leftarrow selectBest(pop)$;
    $children \leftarrow applyCrossover(parents)$;
    $applyMutation(children)$;
    $gen \leftarrow gen + 1$;
end for
return $bestIndividual$;

2.1.5 Position and Velocity Update

2.1.6 Parameters

Talk about Adjusted and Standardized fitness (also hits, and about human interpretation)

2.2 Cooperative Particle Swarm Optimization

et al. [1]

2.2.1 Variant: CPSO-S

¡Include algorithms¿

2.2.2 Variant: CPSO-Sk

¡Include algorithms¿

2.2.3 Variant: CPSO-Hk

¡Include algorithms¿

2.2.4 Variant: CPSO-Rk

¡Include algorithms¿

2.3 Delaunay Triangulation

¡discuss the creation as well as the usages¿ ¡include the pseudocode for n-dimensions¿

3 Literature Review

¡discuss the cpso papers and the delaunay triangulation papers¿

$$addgeneralGPsymbolicregressionhere(nofinance)$$

4 Experiments

4.1 Experiment 1: CPSO-S

4.1.1 Setup

4.1.2 Results

4.1.3 Discussion

system_design.png

Figure 1: Design of the GPP System.

4.2 Experiment 2: CPSO-Sk

4.2.1 Setup

4.2.2 Results

4.2.3 Discussion

4.3 Experiment 3: CPSO-Hk

4.3.1 Setup

4.3.2 Results

4.3.3 Discussion

4.4 Experiment 4: CPSO-Rk

4.4.1 Setup

4.4.2 Results

4.4.3 Discussion

5 Conclusion

A Preliminary Trials

References

[1] Kalyan Veeramachaneni, Owen Derby, Una-May O'Reilly, and Dylan Sherry. Learning regression ensembles with genetic programming at scale. *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 1117–1124, 2013.