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 [1]. 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. The initial position of the particle should be randomly generated to allow the swarm to effectively test the entire search space. If they all begin with similar solutions, it is unlikely that all potential

solutions are tested which leads to a higher possibility of converging on a local minimum as opposed to the best overall value. 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 each iteration with the following functions.

Position Update Formula

$$x_i = x_i + v_i$$

Updating the position of the particle is as simple as taking the current position and adding the current velocity to it. For multi-dimensional problems where the particles have multiple dimensions, this addition will need to be done for each dimension. In this case, the i represents the index of the position.

Velocity Update Formula

$$v_i = \alpha v_i + \omega C_1(y_i - x_i) + \lambda C_2(\hat{y}_i - x_i)$$

The velocity update function has a little more to it. The new velocity is determined by a combination of it's current velocity, the distance it is from the particles personal best and the distance it is from the global best. In this case y_i represents the personal best for that index and $\hat{y}_i$ represents the global best. The influence that each has is determined by the inertial weight (α), the cognitive weight (ω) and social weight(λ) respectively. These values typically add up to 1 and determine the percentage of influence they have on the new velocity. Additionally, the formula also makes use of a random component. The values C_1

and C_2 are randomly generated values in the range [0,1] that adds more randomness and aids in keeping the swarm from converging too quickly.

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 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 as seen in the velocity update formula above. This transfer of knowledge amongst the particles helps the swarm converge on the optimal solution. With each iteration, the swarm tries a number of different potential solutions and gains information to help it decide the overall best solution. The code for the swarm is relatively straight forward.

Table 1: PSO Pseudo Code

```
Initialize particles randomly
while current iteration < max iterations do
    for each particle j ∈ swarm do
        //Update the personal and global bests
        if f(j.x⃗) > f(j.y⃗) then
            j.(y⃗) ← j.(x⃗)
        end if
        if f(j.y⃗) > f(ŷ) then
            (ŷ) ← j.(y⃗)
        end if
    end for
    for all particles ∈ swarm do
        Update the Velocity
        Update the Position
    end for
end while
```

For every iteration, all the swarm needs to do it update the values of the personal best for each particle (represented by $\vec{y}$) and update the global best if a new best value is

found. This is accomplished by passing the solution set into the fitness function (see below) and getting a value. If you are working on a minimization problem, the smaller of the two will become the personal best, otherwise you will take the larger value. It then needs to iterate through each particle updating it's velocity and position. It is important to complete the global best update before updating the velocity since the global best has an influence on the new velocity. If the global best changes during an iteration, that new global best needs to be reflected in the velocity update in ALL particles, not just the particles that appear afterwards.

2.1.4 Fitness Function

In order to find the best (or optimal) solution to a problem, we first need a way to evaluate those solutions. The Fitness Function is responsible for taking the list of inputs that make up the solution and determining how well it fits the problem. It will then return a single value that helps the swarm determine if it is a good solution or a bad solution, ultimately helping it converge on the best value. The Fitness Function itself is the representation of the problem that we are looking to optimize and is how we tailor the Particle Swarm Optimization algorithm to solve different problems.

2.1.5 Parameters

There are a number of different parameters that can be altered to affect the success of the algorithm. Increasing or decreasing these values can lead to different results depending on the problem at hand. Experimentation is often required in order to get the best performance out of the algorithm.

Max Iterations: This helps determine how many iterations the swarm will go through in to search for the optimal solution. Increasing this value can potentially lead to better results however is meaningless if the swarm were to have already con-

verged to a local maxima.

Swarm Size: the size of the swarm determines the number of particles it employs to search for the optimal solution. Increasing this gives the swarm more information for each iteration and allows the swarm to have a higher likely hood of finding the overall best solution

Inertial Weight (α): the Inertial weight determines how much of an influence the current velocity has on future velocities. A large Inertial weight will lead to the particle carry on in it's current direction allowing it to better search the search space (though potentially heading off towards worse values). A low Inertial weight will allow it to converge towards the best values more quickly but may have a hard time searching the rest of the potential values

Cognitive Weight (ω): the Cognitive weight notes the effect the particles personal best solution has on the future velocity of the particle. Increasing this will keep the particle from venturing too far from this solution and only explore the neighboring space.

Social Weight (λ): this is called the social weight because it determines the effect other particles have on the current particle. This is the influence the global best (or the best solution currently found by the entire swarm) has on the future velocity of each particle. A low value will have the particles mostly work on their own, whereas a large social weight will have all particles converge toward the best value.

2.2 Cooperative Particle Swarm Optimization

et al. [2]

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¿

$$addgeneralGPsymbolicregressionhere(nofinance)$$

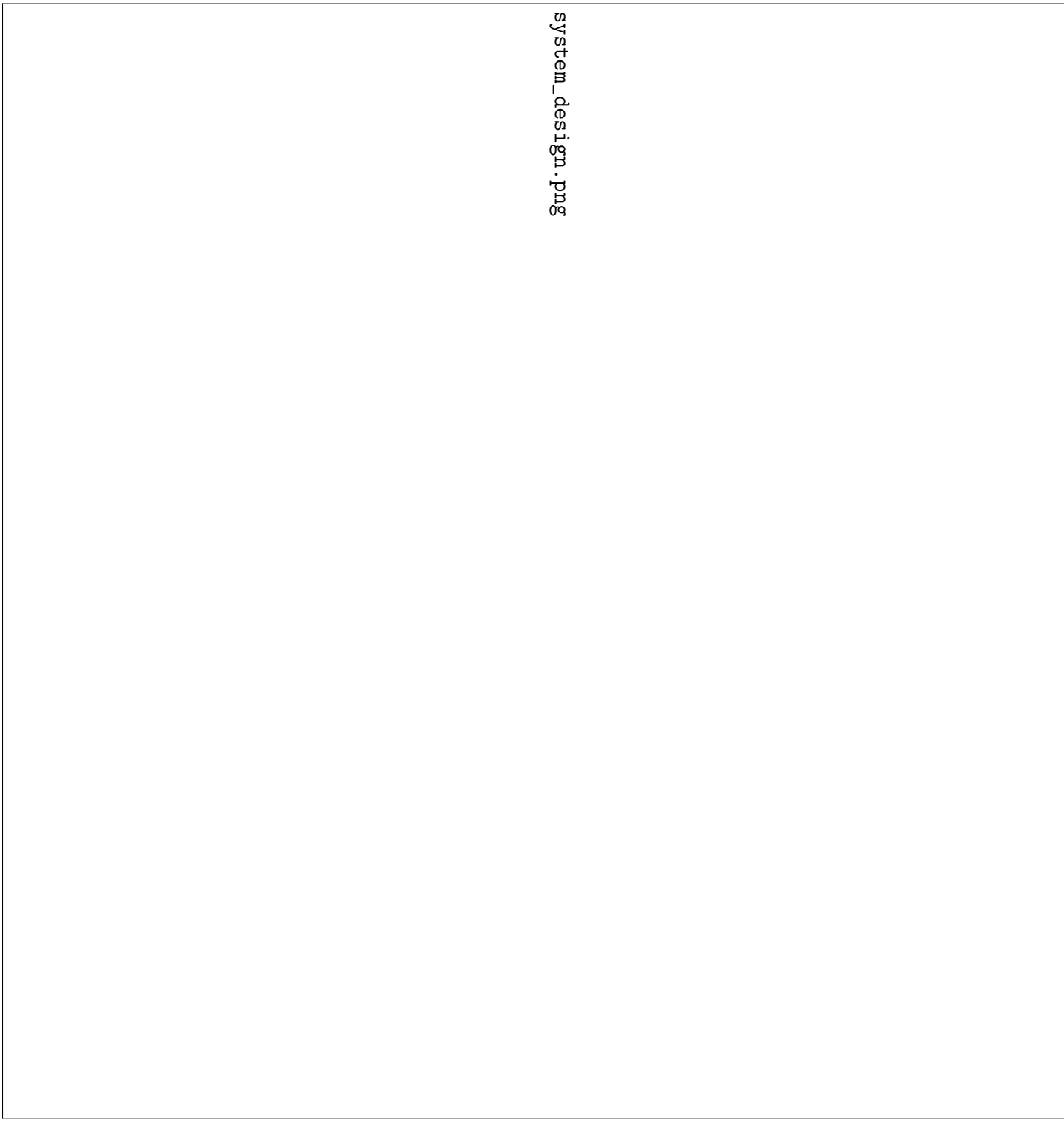

Figure 1: Design of the GPP System.

4 Experiments

4.1 Experiment 1: CPSO-S

4.1.1 Setup

4.1.2 Results

4.1.3 Discussion

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] Alan Mackworth David Poole and Randy Goebel. Computational intelligence: A logical approach. pages 1–21, 19986.

[2] 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.