Comparison and Implementation of PSO Algorithm Variants to Localize Gravitational Wave Source Signals

Elliot Eckholm
Department of Astronomy and Astrophysics
University of California Santa Cruz
Email: eeckholm@ucsc.edu

Project Supervisor: Soumya Mohanty
Department of Physics and Astronomy
University of Texas Rio Grande Valley
Email: Soumya.mohanty@utrgv.edu

Precisely localizing a gravitational wave (GW) source is one of the biggest challenges in GW astronomy. Once a GW signal is detected and confirmed by multiple detectors, a coherent network analysis is done using parameters, such as the time delay, to determine an approximation of the source location. The problem is that it is simply too computationally expensive to find a precise location quickly. PSO algorithms have been established for working reliably and efficiently in helping to locate a GW source to high precision[3], so this paper will explore which variation of PSO should be used. From our results, it seems the lBest PSO and standard PSO meet our needs.

1 Introduction

Particle Swarm Optimization (PSO) algorithms, have garnered a lot of attention in recent decades for their wide-ranging applicability, flexibility and simplicity. PSO is a stochastic algorithm that is based on the behavioral properties of swarms. Since the first developed PSO there has been a lot of focus on designing the best performing PSO and thus many variants have been produced over the years[1]. In this paper, we will discuss and compare three of the most popular variants of PSO: gBest, lBest, and standard PSO (sPSO). We will also explain how to implement sPSO into one's own project and be used on any desired fitness function so it can be applied to the Gravitational Wave localization problem.

The general PSO algorithm works by having virtual particles that make up a swarm, fly around a predefined search space and "look" for the most optimal location. If you think of each particle as a bee, then the most optimal location would be the location where there is the highest concentration of flowers for example, and the field of flowers in which the bees fly in would be called the search space. Each bee constantly checks their current location and calculates how "good" that location is. "Good" in this case means the higher the concentration of flowers, the better the location. So each bee keeps track of it's personal best location it has found so far on it's own, and is able to constantly compare it's personal best location with it's current location to see if it is better. The bee also compares it's personal best with the swarm's best location. The swarm's best location is simply the best of all the personal best locations found by any bee. If a bee is flying around and happens to find a new personal best location, and then compares this new personal best location to the swarm's best location, and the personal best is better, then the swarm's best location is replaced by this bee's new personal best. So at any given time, one bee's personal best is also the swarm's best. The bee's keep flying around searching for a better location until the swarm's best has reach a certain criterion, such as a maximum number of iterations. PSO can be generalized to any number of dimensions, not just the four dimensions that bees are limited to, and can be applied to any problem that requires finding an optimal solution simply by changing the definition of what a "better location" means. This calculation of "better" is done by a fitness function. This process is illustrated in Figure 1[1] where after 200 iterations, the particles swarm around the global optimum.

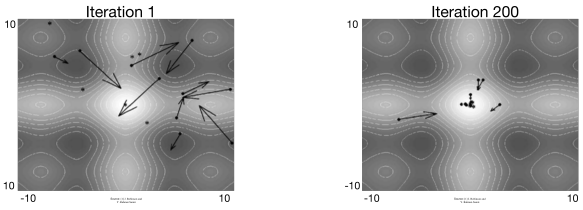

Fig. 1. After 200 iterations the particles are swarming in on the optimum.

Nomenclature

Particle/Agents: An individual within the swarm.

Swarm: The entire group of particles.

Search Space: The N-dimensional space in which the optimal location is contained.

Position/Location: An N-dimensional coordinate within the search space

Fitness: A single number that represents how good a location is.

pBest: A location in the search space that has the best fitness found my a particle so far.

gBest: A location in the search space that has the global fitness found by any particle in the entire swarm so far.

2 General Particle Movement

The particles fly around using the displacement equation (1) and position equation (2)[1]. Equation (1) is the sum of three vectors where the first vector is the particle's weight, which provides resistance to be being "pulled" toward the rest of the swarm. The second vector is the particle's displacement toward it's personal best location while the third vector is a displacement towards the swarms best location. This all results in the particle being attracted towards the swarm while still exploring around the search space.

$$\begin{aligned} v_n = {} & w * v_n \\ & + c_1 * rand() * (p_{best} - x_n) \\ & + c_2 * rand() * (g_{best} - x_n) \end{aligned} \tag{1}$$

$$x_n = x_n + v_n \tag{2}$$

3 Overview of gBest, lBest and standard PSO

The following sub-sections will outline each algorithm as well as explain the advantages and disadvantages to each.

Differences in PSO Variations			
PSO Variation	gBest	lBest	sPSO
Synchronous	Yes	Yes	No
Ring Topology	No	Yes	Yes
Randomized Iterations	No	No	Yes
Unbiased coordinate displacement	No	No	Yes
Invisible Walls	Yes	Yes	No
Discrete Search Space	Yes	Yes	Yes
Random Ring Topology	No	No	Yes

3.1 gBest PSO

gBest PSO is the simplest of the variants in this paper. It works by just having each particle's fitness compared with the swarm's best, or global best, hence the name gBest. It works very well for low dimensional problems but decreases in performances rapidly above dimensionality > 5. It is however, fast for these lower dimensional problems, so it is best to use gBest in simple problems where speed is important. Here is the general outline for the gBest Algorithm:

1. Define Search Space (i.e set min and max boundaries in each dimension)
2. Initialize swarm with random initial positions and velocities inside the search space
3. FOR each Iteration:
   (a) FOR each Particle:
      i. IF Particle is outside search space: do not evaluate fitness
      ii. ELSE: Evaluate Fitness
      iii. IF Fitness of Particle's current position $<$ Fitness of gBest:
      gBest = Particle's current position
      iv. IF Fitness of Particle's current position $<$ Fitness of Personal Best:
      Personal Best = Particle's current position
      v. Update Particle's velocity
      vi. Update Particle's position

3.2 lBest PSO

lBest is considered the best of both worlds between gBest and standard PSO[2]. It is more robust than gBest because it uses the ring topology and finds the best fitness within a neighborhood of particle's instead of the entire swarm. Then each neighborhood can communicate their local best with each other, hence the name lBest. However it is less customizable than standard PSO. lBest seems to work very well for both low and high dimensional problems. Here is the general outline for the lBest Algorithm:

1. Define Search Space (i.e set min and max boundaries in each dimension)
2. Initialize swarm with random initial positions and velocities inside the search space
3. FOR each Iteration:
   (a) FOR each Particle:
      i. IF Particle is outside search space: do not evaluate fitness
      ii. ELSE: Evaluate Fitness
      iii. Compare current Particle's fitness with adjecent Particles' Fitnesses
      iv. Find particle with best fitness in this neighborhood
      v. IF Fitness of particle with best fitness in the neighborhood $<$ Fitness of lBest:
      lBest = This Particle's position
      vi. IF Fitness of Particle's current position $<$ Fitness of Personal Best:

Personal Best = Particle's current position

vii. IF Fitness lBest $<$ Fitness of gBest:
gBest = lBest

viii. Update Particle's velocity

ix. Update Particle's position

3.3 Standard PSO

Since so many PSO variants have been developed, standard PSO was made in order to establish a PSO that could be used as the foundation for comparisons. It is the more robust and customizable PSO in this paper, and thus has many advantages. Like lBest, it uses the concept of finding the neighborhood best, but it selects random particle's in the swarm to compose a neighborhood instead of simply the adjacent particles. sPSO also randomized the order in which particles are iterated through and updated in the swarm which similar to how to a swarm actually behaves. However, even with the increases robustness sPSO and lBest performances are similar at mid to low dimensions such as 20. But at high dimensionality (ie greater than 30) sPSO seems to out perform lBest in terms of convergence but can take longer computationally depending on the fitness function. Here is the general outline for the sPSO Algorithm:

1. Define Search Space (i.e set min and max boundaries in each dimension)
2. Initialize swarm with random initial positions and velocities inside the search space
3. FOR each Iteration:

   (a) FOR each Particle: Randomize the order of updating the Particles for each iteration

   i. IF Particle is outside search space: set Particle's velocity = -0.5 Particle's velocity

   ii. ELSE: Evaluate Fitness

   iii. Compare current Particle's fitness with 2 other randomly selected Particles' Fitnesses

   iv. Find particle with best fitness in this group of 3 particles

   v. IF Fitness of particle with best fitness in the group $<$ Fitness of lBest:
   lBest = This Particle's position

   vi. IF Fitness of Particle's current position $<$ Fitness of Personal Best:
   Personal Best = Particle's current position

   IF Fitness lBest $<$ Fitness of gBest:
   gBest = lBest

   vii. Update Particle's velocity based on the gravitational center

   viii. Update Particle's position on the gravitational center

4 Comparison on Benchmark Functions

To illustrate the differences in performance of each PSO algorithm, we ran gBest, lBest and sPSO on benchmark functions (1) and (2) over four different dimensions (5, 10, 20, and 30) that each had an optimal location at the origin. I ran each PSO for 3000 iterations.

$$Griewank = (\frac{1}{4000})\sum_{i=1}^{Dim} x_i^2 - \prod_{i=1}^{Dim} cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (3)$$

$$Rastrigin = \sum_{i=1}^{Dim} (x_i^2 - 10cos(2\pi x_i) + 10) \quad (4)$$

5 Results

This sections contains the graphs and analysis of gBest PSO, lBest PSO and sPSO on the Griewank and Rastrigin functions at various dimensions.

5.1 gBest PSO Analysis

As you can see in Figures 2 and 3 below, gBest does not converge for dimensions > 5. In other words, the particles never find a location with a better fitness value than the ones they initially start with. This leads to the conclusion that the gBest PSO is not well suited for high dimensional problems. But for lower dimensional problems (< 5) gBest converges in less than 10 iterations for both benchmark functions. This makes gBest ideal for simple problems where minimizing computationally cost is important.

Fig. 2. gBest PSO used with Griewank Benchmark function on 5, 10, 20 and 30 dimensions.

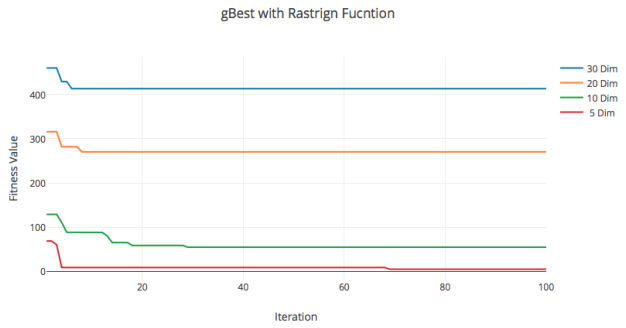

Fig. 3. gBest PSO used with Rastrigin Benchmark function on 5, 10, 20 and 30 dimensions.

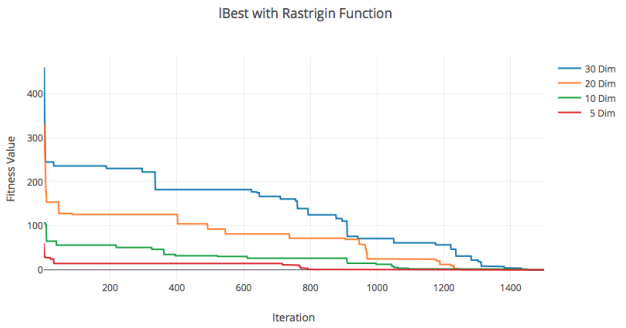

Fig. 5. lBest PSO used with Rastrigin Benchmark function on 5, 10, 20 and 30 dimensions.

5.2 lBest PSO Analysis

Overall the lBest PSO performs well across all four dimensions on both benchmark functions as you can see in Figures 3 and 4. This make lBest an acceptable solution to many different problems with varying complexity and dimensionality, including localizing Gravitational Waves. One downside to the lBest PSO is, for the Griewank benchmark function, the convergence rate is very fast, meaning there is not a lot of explorations happening. This can lead to the particles swarming a local optimum instead of the global optimum. In the case for the Griewank benchmark this fast convergence could also be due to the simplicity of the benchmark function itself in that it has an inherently quicker convergence.

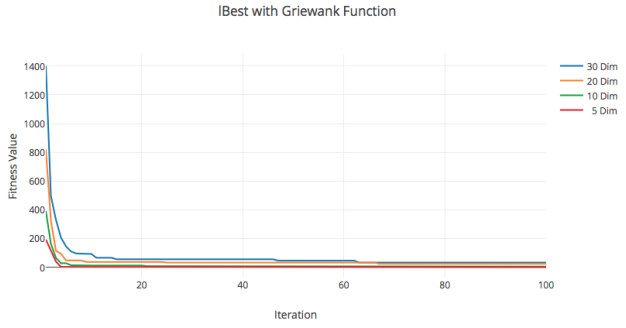

Fig. 4. lBest PSO used with Griewank Benchmark function on 5, 10, 20 and 30 dimensions.

5.3 sPSO Analysis

The Standard PSO also performs very well for both benchmark functions and at all four dimensions. As mentioned in the lBest PSO analysis, sPSO seems to have a fast convergence for Griewank benchmark function as well, supporting the claim that this benchmark function simply has an inherently quicker convergence. sPSO has the largest exploration phase of the three algorithms. This is due to sPSO's decrease in influence of the swarm's best location on each particle thus particles take longer to converge. Overall, sPSO has a solid balance of exploration and a reasonable convergence time, thus making it an acceptable PSO algorithm for the Gravitational Wave localization problem. It also is the most robust of the three algorithms and thus can be customized to be even better suited to a specific problem.

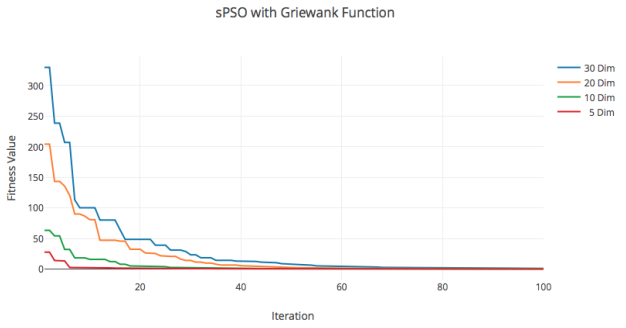

Fig. 6. sPSO used with Griewank Benchmark function on 5, 10, 20 and 30 dimensions.

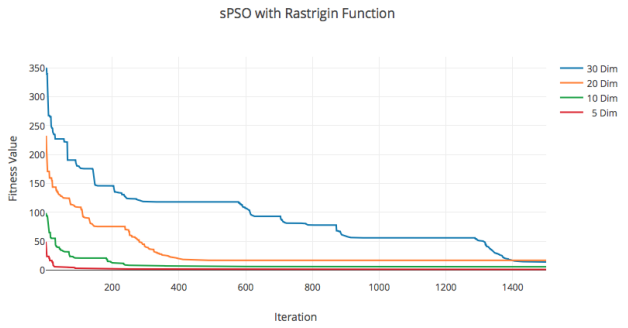

Fig. 7. sPSO used with Rastrigin Benchmark function on 5, 10, 20 and 30 dimensions.

6 Conclusion

From the analysis' of gBest PSO, lBest PSO and standard PSO, we can conclude the following:

1. All three PSOs' rates of convergence are directly proportional to dimensionality
2. For dim < 5 gBest PSO converges quickly to the optimum but does not converge in higher dimensions for either benchmark functions, thus gBest is best suited for simpler problems.
3. For dim > 20 lBest PSO has the highest rate of converges for both benchmark functions, meaning lBest works well for all four dimensions tested.
4. sPSO has a larger exploration phase for both benchmark functions and thus allows the particles to explore the entire search space but still converges at a reasonable rate for all dimensions tested.

From these results, it seems the lBest PSO and standard PSO will best meet our needs.

7 sPSO Implementation

The overall goal with this integration was to be able to call the sPSO Algorithm from another project and run it with a fitness function that you already have. That way all we have to do is call the sPSO code from the LIGO Data Analysis project. The problem was that sPSO code was not written to be used in this way, it was a self contained project that used it's own predefined fitness functions. We generalized the sPSO code and added a level of abstraction so that any fitness function could be used. We also created a wrapper function that setups the sPSO algorithms and runs it correctly. With these modifications, all one has to do to integrate sPSO into their project is import the source files and run the wrapper function on the desired fitness function. All of this code, including an example project of sPSO successfully integrated as well as a manual to the code integration can be found at:
`https://github.com/ElliotEckholm/Integrated_sPSO`

7.1 List of Modifications to Original sPSO

1. Removed all definitions of predefined fitness functions
2. Deleted unnecessary source files that contained functions specific to certain fitness functions
3. Switched to the Native C random number generator, the KISS rng was giving me odd errors
4. Changed input of sPSO function to use a pointer function
5. Added several new inputs into sPSO so that the PSO param struct, PSO results struct, and fitness function parameter struct information could be accessed throughout the code
6. Instead of having switch statements for each fitness function and their corresponding parameters, I made pointer functions and structs that stored the information. This added a new level of abstraction so that any kind of fitness function could be called as long as it was in the correct format
7. Had to allocate GSL vectors and convert GSL vector into doubles throughout various functions in the sPSO code
8. Placed all sPSO setup code into the wrapper sPSO function and called sPSO inside here so that sPSO could be called easily by simply calling the wrapper sPSO function from within Mohanty code
9. Had to convert all inputs from Mohanty code into the format sPSO required
10. Had to convert all outputs from sPSO back to the format Mohanty code required

Acknowledgements

This project was supported by the National Science Foundation under Grant No. 1461237. I would like to thank Soumya Mohanty for all of the time he has dedicated to guiding me through this project. I would also like to thank Marc Normandin for his code and guidance.

References

[1] J. Robinson and Y. Rahmat-Samii, "Particle Swarm Optimization in Electromagnetics," in IEEE Transactions on Antennas and Propagation, Vol. 52, No. 2, Feburary 2004

[2] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in Proceedings of the 2007 IEEE Swarm Intelligence Symposium

[3] S. Weerathunga and S. Mohanty, "Performance of Particle Swarm Optimization on the fully-coherent all-sky search for gravitational waves from compact binary coalescences," University of Texas Rio Grande Valley,, TX, 2017