Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 20 (2013) 71 – 76

Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

The Genetic Flock Algorithm

Jeffrey Brooks[a], David Hibler[b]*

[a,b]ChristopherNwport University, PCSE Department, 1 University Place, Newport News, Virginia 23606, USA

Abstract

The purpose of this paper is to describe and evaluate a new algorithm for optimization. The new algorithm is named the Genetic Flock Algorithm. This algorithm is a type of hybrid of a Genetic Algorithm and a Particle Swarm Optimization Algorithm. The paper discusses strengths and weaknesses of these two algorithms. It then explains how the Genetic Flock Algorithm combines features of both and gives details of the algorithm. All three algorithms are compared using eight standard optimization problems that are used in the literature. It is shown that the Genetic Flock Algorithm provides superior performance on 75% of the tested cases. In the remaining 25% of the cases it outperforms either the Genetic Algorithm or the Particle Swarm Optimization Algorithm; it is never worse than both. Possible future improvements to the Genetic Flock Algorithm are briefly described.

Selection and peer-review under responsibility of Missouri University of Science and Technology
Keywords: Particle Swarm Optimization; Genetic Algorithm; Optimization

1. Introduction

This paper describes the Genetic Flock algorithm. This algorithm is a hybrid between a Genetic Algorithm and a Particle Swarm Optimization Algorithm. An evaluation of all three algorithms on some standard optimization problems is given.

Popularized by John Holland in the mid-1970's [1], the Genetic Algorithm (GA), ties together biologically inspired theories of evolution, genetics, natural selection, and mutation into a computer algorithm that heuristically searches for an optimal solution to a problem. A possible solution to a problem is characterized by a genome that normally consists of a single chromosome. The chromosome contains values called genes. Mutation and crossover are used to change chromosomes. Mutation consists of random small changes to chromosomes. Crossover involves swapping large amounts of genetic material.

In a GA a population of chromosomes is evolved in the following way. The population is randomly initialized. New generations are produced by repeatedly selecting the fittest individuals and subjecting them to mutation and crossover. This process continues until one of the chromosomes has satisfactory fitness or until a limit on the

* Corresponding author. Tel.: +1-757-594-7360; fax: +1-757-594-7919.
E-mail address: dhibler@cnu.edu.

Selection and peer-review under responsibility of Missouri University of Science and Technology
doi:10.1016/j.procs.2013.09.241

number of generations is reached. Details and variations of this algorithm are discussed in many places, for example in Engelbrecht [2].

More recently, James Kennedy and Russell Eberhart developed a fascinating new heuristic optimization algorithm they called Particle Swarm Optimization [3]. Details and modern variations of this algorithm are also given in Engelbrecht [2]. Like the Genetic Algorithm, this algorithm is inspired by biological phenomena.

In Particle Swarm Optimization (PSO), a swarm of particles is used. Each particle represents a potential solution to the given optimization problem. The particles are "flown" in discrete time steps through a multidimensional search space where the position of each particle is adjusted according to its own experience and that of its neighbors to find the best value of a given function.

In the second section of this paper, we describe the Genetic Flock Algorithm. In the third section we compare all three algorithms using eight standard optimization problems that appear in the literature. In the final section we give our conclusions and possible future improvements to the Genetic Flock Algorithm, (GFA).

2. The genetic flock algorithm

2.1. Comparison of GA and PSO

2.1.1. Similarities

There is an obvious correspondence between a GA and a PSO that makes a hybrid algorithm possible. They both consist of groups of potential solutions to the problem of interest. Usually chromosomes are bit strings; however, they can be arrays of real numbers. A particle in a PSO, on the other hand, has a location in a high dimensional space, which is usually a collection of real numbers. With some modification, the location can be given by a collection of bits [2]. In this paper, we will consider both chromosomes and particle locations as arrays of real numbers. Chromosomes correspond to particle positions.

Since chromosomes change in each generation, these generations correspond to the discrete time steps in which particle locations change.

The fitness function of a Genetic Algorithm corresponds to the function being optimized in the PSO Algorithm.

2.1.2. Differences

The essential difference between the two algorithms lies in the way the individuals change. Chromosomes change by random mutation and crossover. Each particle changes by random oscillation about two centers of attraction. One of these centers is the particle's best position so far. This is sometimes called the *cognitive component* since it is based on the experiential knowledge of the particle [2]. The other center for a given particle is the best position found by any particle in the given particle's neighborhood. This is sometimes called the *social component* since it is based on socially exchanged information. If a particle's neighborhood is the entire swarm, the method is called *global best*. If the particle's neighborhood is more restricted, the method is called *local best*.

It is immediately apparent that GAs are more random than PSOs. If we identify parents with children, a given chromosome may not be improved by mutation and crossover. It may have a worse fitness. The fitness of the best chromosome can even decline from generation to generation. For this reason, genetic algorithms usually use elitism. This method requires the best few individuals in a generation to be preserved unchanged into the next generation. PSO's have no need for elitism as previous bests are never lost.

In some ways PSO's are like more sophisticated versions of hill climbing. The previously found values strongly influence the search. In GAs the previous values are only a base from which changes are made. They do not affect the nature of the changes. Crossover in GAs may combine partial solutions, but it also sometimes disrupts them.

Our expectation is that a PSO should be faster than a Genetic Algorithm, but less robust in that it may be trapped by a sufficiently irregular function to optimize.

2.1.3. Genetic Flock

It is our belief that an integration of PSO and GA into each individual step of an algorithm is unnecessary and, in fact, may be detrimental. Instead, we have developed a hybrid of a Genetic Algorithm and a PSO which has separate PSO and GA phases. We called this a Genetic Flock Algorithm. The main contribution of this paper is to show the advantages of such a hybrid method with separate GA and PSO phases. This is achieved by comparing it with the individual algorithms.

The Genetic Flock works by executing the Genetic Algorithm phase for a fixed number of iterations and then executing the Particle Swarm Optimization phase for a fixed number of iterations. The number of iterations in each phase may be adjusted, but the default is 50 for each. This process is repeated until the maximum number of iterations has been reached.

The GA phase uses tournament selection with single point crossover. Mutations are additive and are chosen from a Gaussian distribution with standard deviation a fixed fraction (typically .01) of the range over which optimization is to occur. The default mutation rate is 0.10.

In the PSO phase, particles are grouped together into collections called neighborhoods with a default size of 7. Particles move to a new location based on inertia, a randomized spring-like force towards the best location the particle visited, and a similar force toward the best location visited by any other member of its neighborhood. Each time the particle moves from one location to another, the spring-like force constants involving the neighborhood and personal best locations are modified by random percentages.

A particle's maximum velocity is not allowed to exceed one half the distance between the particles' boundaries; that is to say, that a particle is not allow to take a single step of a size that was bigger than one half of the entire space being searched.

The algorithm can be set to run for a fixed number of generations or until there is insufficient progress.

2.1.4. Other work

Eberhart and Shi [4] examine the similarities between the Genetic Algorithm and Particle Swarm Optimization Algorithm, and emphasize that elements from one algorithm should be incorporated into the other. In line with that recommendation, Castelli, Manzoni, and Vanneschi [5] reference authors who have incorporated the Particle Swarm Optimization element of memory into the Genetic Algorithm. Likewise, Engelbrecht [2] mentions authors who have integrated mutation into the Particle Swarm Optimization algorithm.

The most similar work to ours is due to Juang [6]. In his work, both the Genetic Algorithm and Particle Swarm Optimization Algorithm are fused into a single algorithm. Juang uses this algorithm to assign weights for a Neural Network (another heuristic algorithm), and evaluates his hybrid algorithm based on the performance of the Neural Network. In his hybrid algorithm, Juang only performs the Particle Swarm Optimization portion on the best performing half of all chromosomes from the Genetic Algorithm – these chromosomes are called "Enhanced Elites". Juang then uses the Enhanced Elites to go through selection, crossover, and mutation. The resulting offspring produce one half of the next generation, with the Enhanced Elites making up the remainder of the next generation.

In our view, even Juang's algorithm is unnecessarily complicated. Our objective in this paper is to show that a simple phased algorithm is already sufficient to compensate for the weaknesses in the GA and PSO methods.

3. Tests

A total of eight test problems were used to evaluate the performance of the three algorithms. These problems span from simple to complex in an attempt to develop a well-rounded picture of how well each algorithm performs. These particular test problems are known benchmark algorithm evaluation problems, which were designed for the purpose of testing the performance of optimization algorithms[7,8,9]. The following test problems were used.

1. Spherical $f(\vec{x}) = \sum_{j=1}^{n} x_j^2$ where $-100 \le x_j \le 100$

2. Ackley $f(x) = -20\exp\left(-.02\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}\right) - \exp\left(\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)\right) + 20 + e$ where $-30 \le x_j \le 30$

3. Griewangk $$f(\vec{x}) = \sum_{j=1}^{n} \frac{x_j^2}{4000} - \prod_{j=1}^{n} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1 \quad \text{where } -500 \le x_j \le 500$$

4. Michalewicz $$f(\vec{x}) = -\sum_{j=1}^{n} \sin(x_j)\left(\sin\left(\frac{jx_j^2}{\pi}\right)\right)^{2m} \quad \text{where } m = 10,\ 0 \le x_j \le \pi$$

5. Rastrigin $$f(\vec{x}) = 10n + \sum_{j=1}^{n}\left(x_j^2 - 10\cos(2\pi x_j)\right) \quad \text{where } -5.12 \le x_j \le 5.12$$

6. RosenBrock $$f(\vec{x}) = \sum_{j=1}^{n/2}\left(100(x_{2j} - x_{2j-1}^2)^2 + (x_{2j-1} - 1)^2\right) \quad \text{where } -2.048 \le x \le 2.048$$

7. Schwefel $$f(\vec{x}) = 418.9829n - \sum_{j=1}^{n} x_j \sin\left(\sqrt{|x_j|}\right) \quad \text{where } -500 \le x_j \le 500$$

8. Shekel $$f(\vec{x}) = \sum_{j=1}^{10}\left[\sum_{i=1}^{4}(x_i - c_{ij})^2 + \beta_j\right] \quad \text{where } -10 \le x_j \le 30$$

where $$\vec{\beta} = \frac{1}{10}\begin{bmatrix} 1 & 2 & 2 & 4 & 4 & 6 & 3 & 7 & 5 & 5 \end{bmatrix}$$

$$c = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7.0 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7.0 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}$$

The goal was to minimize f. The minimum value was zero for each f. In each case, a value of n = 10 was used in the above equations.

Population size was chosen to be 56 based on some trial runs. Since n was 10, this represented chromosome size and also the number of PSO dimensions.

In order to generate comparison numbers for the algorithms, each algorithm was run 50 times on each of the test problems. Each run of the algorithm on a given problem consisted of 50,000 steps. After each run, the algorithm's best performing element was recorded. Then for each algorithm, these values were averaged across all runs. This approach allowed for analysis of the average performance of each algorithm on each test problem.

3.1. Typical Dynamic Behavior

Runs that show typical time behavior are discussed below. Due to space limitations only a brief description of results in each case is given. For more details see Brooks [10].

For the Spherical test problem the Particle Swarm Optimization Algorithm got off to a quick start and built a lead that the other two algorithms were not able to overcome. The Genetic Flock Algorithm realized most of its improvements while it was operating in the Particle Swarm Optimization mode, which allowed it to distance itself from the Genetic Algorithm. Both the Genetic Algorithm and Genetic Flock algorithms continued to improve throughout their iterations; however, neither was able to match the pace set by the Particle Swarm Optimization algorithm.

For the Ackley problem, the Particle Swarm Optimization Algorithm got off to a rapid start; however, it was soon trapped in a local minima and stopped improving. This can occur if the entire population gets bound within a local minimum, and the best locations (both neighborhood and personal) all fall within the local minimum. The other two algorithms continued to show progress through their iterations, with the Genetic Flock Algorithm realizing faster

progress than the Genetic Algorithm. Both algorithms produced a lower result than the Particle Swarm Optimization Algorithm, with the Genetic Flock Algorithm finding the lowest result of the three algorithms.

On the Griewangk equation, the Particle Swarm Optimization algorithm jumped out to an early lead. The Genetic Flock algorithm was able to close the distance between the two when it switched to Particle Swarm Optimization mode. Later, it was able to overtake the Particle Swarm Optimization Algorithm. The Genetic Algorithm showed continual improvement, however it was not able to catch either of the other algorithms.

The Particle Swarm Optimization algorithm performed poorly in comparison to the Genetic Flock and the Genetic Algorithm on the Michalewicz problem. While the Particle Swarm Optimization Algorithm continued to improve over time, the other two algorithms did so much faster. The Genetic Flock and Genetic Algorithm remained close initially, but the Genetic Flock Algorithm was able to stay in front and reach a lower value than the Genetic Algorithm.

The results for the Rastrigin equation mirrored that of the Michalewicz equation. The Particle Swarm Optimization improved throughout its iterations; however, its improvement was noticeably slower than the other two algorithms. Between the Genetic Algorithm and Genetic Flock, the Genetic Flock was able to reach a lower average value.

For the RosenBrock equation, the Particle Swarm Optimization Algorithm obtained an early lead. The Genetic Algorithm and Genetic Flock Algorithm looked similar in the early iterations. Eventually the Genetic Flock was able to catch up and outperform the Particle Swarm Optimization Algorithm. The Genetic Algorithm had a steady improvement trajectory; however it was not able to keep pace with the other two algorithms.

The performance of the Particle Swarm Optimization Algorithm was significantly worse than the performance of the other algorithms on the Schwefel problem. To understand why this was so, subsequent executions were run with a smaller range of possible values, under which the Particle Swarm Optimization was able to perform considerably better.

The performance of all three algorithms was very close for the Shekel equation. The Particle Swarm Optimization Algorithm started the slowest and the Genetic Flock Algorithm separated itself as the best performing algorithm when it switched from the Genetic Algorithm mode to Particle Swarm Optimization mode. After which, the Genetic Flock displayed the best performance. The Genetic Algorithm and Particle Swarm Optimization Algorithm switched places in the early going, but as the iterations progressed the Genetic Algorithm was able to outperform the Particle Swarm Optimization.

3.2. Overall Comparison

The collective results can be considered in the following table which records the lowest averaged value from all repetitions that the algorithms returned. The comparison of averaged values designates which algorithm will perform better on average. To make the table easier to interpret, the best values are in bold and worst values are in italics.

Table 1. Lowest Averaged Result For All Test Problems

Function	Genetic Algorithm	Genetic Flock	PSO
Spherical	*1.74 x 10^{-5}*	2.00 x 10^{-19}	**0**
Ackley	1.56 x 10^{-3}	**1.40 x 10^{-10}**	*6.67 x10^{-1}*
Griewangk	*5.45 x 10^{-2}*	**5.90 x 10^{-3}**	4.30 x10^{-2}
Michalewicz	4.89 x 10^{-7}	**0**	*4.06 x 10^{-1}*
Rastrigin	8.66 x 10^{-6}	**1.71 x 10^{-15}**	*4.29*
RosenBrock	*7.73 x 10^{-3}*	**9.04 x 10^{-15}**	3.66 x 10^{-3}
Schwefel	5.38 x 10^{-3}	**1.27 x 10^{-4}**	*1.16 x10^{3}*
Shekel	**2.21 x 10^{-1}**	**2.21 x 10^{-1}**	*2.23 x10^{-1}*

The Genetic Flock Algorithm had the best averaged performance on six of the eight test problems; it tied for best on another, and had the second best performance on the remaining test problem. The Genetic Algorithm tied the top averaged performance on one problem, was second best on four of the equations, and third on the remaining three equations. The Particle Swarm Optimization Algorithm had the best averaged performance on one problem, second best on two problems, and third on the remaining five problems. Overall these results confirm our expectations concerning these three algorithms.

4. Conclusions

Taking the results as a whole, it becomes evident that for the given test problems the Genetic Flock is the best performing algorithm. In the final tally, the Genetic Flock algorithm showed top performance in 75% of the measurements. Of the remaining 25%, the Genetic Flock outperformed at least one other algorithm. The Genetic Flock was the only algorithm to have no instances of being the worst performing algorithm.

True to their typical characteristics, the Particle Swarm Optimization was very fast; however, it often prematurely converged onto local minima instead of finding the global minimum. The Genetic Algorithm was not as likely to get stuck in local minima; however, it showed slow improvement in the direction of the global minimum. Yet, when the two algorithms acted in unity, neither weakness became as evident as it did when run in isolation.

The Genetic Flock's success can be attributed to its ability to realize the best characteristics from each of its component algorithms. The results show that the Genetic Flock algorithm's component algorithms worked together to compensate for each other's weaknesses. This cooperation was able to produce a better result than the component algorithms achieved on their own.

There are several possibilities for future work. An analysis of performance in terms of median and standard deviation would be useful in order to characterize the behavior of the algorithm in more detail. One possibility for changing the algorithm would be to make the phases of the algorithm dynamic instead of static. Switching between phases could be controlled by the absence of sufficient progress in the current phase. Although the benefit of separate phases has been shown, a comparison with a more integrated hybrid might prove informative. Finally, each separate phase could be fine tuned in order to improve performance.

References

1. Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.
2. Engelbrecht, A. P. (2007). *Computational Intelligence.* West Sussex, England: John Wiley & Sons, Ltd.
3. Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *IEEE*.
4. Eberhart, R., & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. *Evolutionary Programming VII: Proc. 7th Ann. Conf. on Evolutionary Programming Conf.* Berlin: Springer-Verlag.
5. Castelli, M., Manzoni, L., & Vanneschi, L. (2011). The effect of Selection from Old Populations in Genetic Algorithms. *GECCO '11 Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*.
6. Juang, C.-F. (2004). A hybrid of Genetic Algorithm and Particls Swarm Optimization for Reccurrent Network Design. *IEEE Transactions on Systems, Man, and Cybernetics, 34*(2).
7. Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs.
8. Hedar, A.-R. (n.d.). *Global Optimization Test Problems*. Retrieved 10 16, 2011, from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm
9. Bersini, H., Dorigo, M., Langerman, S., Geront, G., & Gambardella, L. (1996, May 20-22). Results of the First International Contest on Evolutionary Optimisation (1st ICEO). *Proceedings of IEEE International Conference on Evolutionary Computation*, 611-615.
10. Brooks, J. (2012). *The Genetic Flock Algorthm.* Master's Thesis, Christopher Newport University