Breeding Swarms: A GA/PSO Hybrid

Matthew Settles[1] and Terence Soule[1]

Department of Computer Science, University of Idaho,
Moscow, Idaho U.S.A 83844

Abstract. In this paper we propose a novel hybrid (GA/PSO) algorithm, Breeding Swarm, combining the strengths of particle swarm optimization with genetic algorithms. The hybrid algorithm combines the standard velocity and update rules of PSOs with the ideas of selection, crossover and mutation from GAs. We propose a new crossover operator (VPAC), incorporating the PSO velocity vector, which actively disperses the population preventing premature convergence. We compare the hybrid algorithm to both the standard GA and PSO models in evolving solutions to four standard function minimization problems. Results show the algorithm to be highly competitive, often outperforming both the GA and PSO.

1 Introduction

Genetic Algorithms (GA) and Particles Swarm Optimization (PSO) are both population based algorithms that have proven to be successful in solving a variety of difficult problems. However, both models have strengths and weaknesses. Comparisons between GAs and PSOs have been performed by Eberhart and Angeline and both conclude that a hybrid of the standard GA and PSO models could lead to further advances [3] [1]. In previous research, a comparison of the two algorithms on recurrent artificial neural networks has led us to the same conclusions [11]. In this paper we present a novel hybrid GA/PSO algorithm, Breeding Swarms (BS), combining the strengths of GAs with those of PSO. The hybrid algorithm is compared to the standard GA and PSO models in evolving solutions to four standard function minimization problems.

The PSO algorithm is conceptually simple and can be implemented in a few lines of code. PSOs also have memory, whereas in a GA if an individual is not selected the information contained by that individual is lost. However, without a selection operator PSOs may waste resources on poor individuals. A PSO's group interactions enhances the search for an optimal solution, whereas GAs have trouble finding an exact solution and are best at reaching a global region [3]. Hybrid GA models are often used to overcome this problem.

Our hybrid algorithm combines the standard velocity and position update rules of PSOs with the ideas of selection and crossover from GAs. The algorithm is designed so that the GA performs a global search and the PSO performs a local search. Other hybrid GA/PSO algorithms have been proposed and tested on function minimization problems [8] and [10]. Lvbjerg incorporated a breeding

(arithmetic crossover) operator into the PSO algorithm, where breeding occurred inline with the standard velocity and position update rules. Results using this implementation yielded little improvement over the standard PSO algorithm. Robinson tested a hybrid which used the GA algorithm to initialize the initial population of the PSO algorithm and another in which the PSO initialized the initial population of the GA. This approach too yielded a small improvement when a GA was used to initialize the PSO algorithm.

In previous work, a GA and PSO were tested and compared when evolving the weights for a fixed topology Recurrent Artificial Neural Network (RANN) [11]. Both the GA and PSO were successful in evolving weights for a RANN that produced a periodic output in response to a fixed input signal. However, the algorithms produced different results for different network sizes. Thus, demonstrating that they implement different search strategies. We believe that the correct combination of both models has the potential to achieve better results faster and to work well across a wide range of problems.

Initial research in designing an algorithm which incorporated elements of both GA and PSO showed promising results [12]. Here the hybrid algorithm was able to outperform both the GA and PSO in evolving the weights for the same fixed topology RANN.

1.1 Genetic Algorithm

Genetic algorithms were first introduced by Holland in the early 1970's [6] and have been widely successful in optimization problems, especially in the binary domain.

In these experiments a real GA, using chromosomes consisting of real values, was chosen for comparison. Tournament selection is used with a tournament size of 3. The crossover operator chosen for use in this experiment was blended crossover (BLX-α) [4]. In blended crossover, two parents are selected using some selection scheme. Each gene in the offspring is then calculated by randomly choosing a position in the range $[min_i - \Delta * \alpha : max_i + \Delta * \alpha]$. Where $min_i = min(x_i, y_i)$, $max_i = max(x_i, y_i)$ and Δ is the distance between x_i and y_i. In this experiment α was chosen to be 0.1.

The mutation operator chosen for this experiment was similar to one used in Bek's masters thesis and shown to work well with real coded GAs [2]. Each individual which is mutated is subject to one of three possible types of mutation defined in equations 1, 2 and 3.

With a probability of 85%

$$gene_{mutated} = gene_{current} + X_{max} * \varphi/3 \tag{1}$$

With a probability of 10%

$$gene_{mutated} = gene_{current} * (M_A)^{\varphi} \tag{2}$$

where M_A is a parameter called mutation altering and φ is a uniform random number in the range [0:1].

With a probability of 5%

$$gene_{mutated} = gene_{current} * \rho \tag{3}$$

Where ρ is a uniform random variable in the range [-1:1].

All tests used a population size of 20. The two best individuals are copied into the next generation (elitism). The mutation rate is 0.5 and the mutation altering parameter was set to 1.5.

The parameters used for each algorithm are shown in 1.

Table 1. Parameters for GA, PSO and BS

Parameter	GA	PSO	BS
Population size	20	20	20
Max iterations	varies	varies	varies
Selection type	elitism & tournament	N/A	tournament
Tournament size	3	N/A	2
Crossover rate	0.8	N/A	N/A
Mutation Rate	0.5	N/A	N/A
Mutation Altering	1.5	N/A	N/A
Social	N/A	2.0	2.0
X_{max}	*varies*	*varies*	*varies*
V_{max}	X_{max}	X_{max}	X_{max}

1.2 The Particle Swarm Optimizer

As described by Eberhart and Kennedy, the PSO algorithm is an adaptive algorithm based on a social-psychological metaphor; a population of individuals (referred to as particles) adapts by returning stochastically toward previously successful regions [7].

During each generation each particle is accelerated toward the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from its previous best position, and the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a set number of times, or until a minimum error is achieved.

The PSO implementation used here is that of Clerc's, PSO with constriction factor.[9]. The governing equation are defined in 4 and 6.

$$\begin{aligned} v_{id}(t) = K[v_{id}(t-1) + c_1 * rand() * (p_{id} - x_{id}(t-1)) \\ + c_2 * rand() * (p_{gd} - x_{id}(t-1))] \end{aligned} \tag{4}$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, where \varphi = c_1 + c_2, \varphi > 4 \quad (5)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (6)$$

v_{id} - The particle i's velocity at dimension d.
x_{id} - The particle i's position at dimension d.
c_1,c_2 - Social parameter.
p_{id} - Particle i's previous best position at dimension d.
p_{gd} - Global best particles position at dimension d.

Clerc, et al., found that by modifying φ, the convergence characteristics of the system can be controlled. For this test φ was set to 4.1, thus $K = 0.73$. Using the recommendations of Shi [13], A limit on the velocity parameter, V_{max}, was set to X_{max}.

1.3 Breeding Swarm

Both Angeline [1] and Eberhart [3] have suggested that a hybrid combination of the GA and PSO models could produce a very effective search strategy. Our goal is to introduce an adjustable hybrid model that makes it possible to optimize the GA/PSO hybrid. Our results show that with the correct combination of GA and PSO, the hybrid can outperform, or perform as well as, both the standard PSO and GA models.

The hybrid algorithm combines the standard velocity and position update rules of PSOs with the ideas of selection, crossover and mutation from GAs. An additional parameter, the breeding ratio, determines the proportion of the population which undergoes breeding in the current generation. The generic hybrid algorithm is as follows.

```
For 1 to Elite
      x <- copy(x_best)

For 1 to (pop_size-Elite)*Breed_Ratio
     x <- Select an Individual
     x <- Update Velocity
     x <- Update Position

For 1 to (pop_size-Elite)*(1-Breed_Ratio)
     x1 <- Select an Individual
     x2 <- Select an Individual
     Crossover(x1, x2)
     Mutate(x1, x2)
```

First, Breeding Swarm copies the n best individuals into a temporary population, Elitism. Next, $(pop\_size - n) * Breed\_Ratio$ individuals are selected, using some selection scheme, to undergo the standard velocity and position update rules of PSO. Third, the remaining individuals needed to fill the population are selected, again by some selection scheme, for crossover and mutation (the selection scheme may or may not be the same scheme as in the previous step). Finally, the temporary population is copied into the working population, fitness is calculated and the process is repeated for some number of iterations. Values for the breeding ratio parameter range from [0.0:1.0] and can be set at either run time (static), stochastic, or adaptive.

In this implementation of Breeding Swarms we chose not to include elitism, elitism of 0, or mutation. The use of the global best individual can be described as a type of elitism and the velocity update rule as mutation. The breeding ratio was set to 0.5. As with any GA the crossover, selection and mutation operators can be modified to suit the problem. Here Tournament selection, with a tournament size of 2, was used to select individuals for velocity and position update and to select parents for crossover. The crossover operator developed for these experiments is a new crossover which incorporates the PSO velocity vector, Velocity Propelled Averaged Crossover (VPAC). The goal is to create two new child particles whose position is between the parents position, but accelerated away from the current direction in order to increase the diversity of the population. Equations 7 and 8 show how the new child position vectors are calculated using VPAC. Towards the end of a typical PSO run, the population tends to be highly concentrated in a small portion of the search space, reducing the effective search space. With the addition of the VPAC crossover operator, a portion of the population is always pushed away from the group, increasing the diversity of the population and the effective search space.

$$c_1(x_i) = (p_1(x_i) + p_2(x_i))/2.0 - \varphi_1 * p_2(v_i) \tag{7}$$

$$c_2(x_i) = (p_1(x_i) + p_2(x_i))/2.0 - \varphi_2 * p_1(v_i) \tag{8}$$

Where φ is a uniform random variable in the range [0.0:1.0]. Positions which fall outside the maximum range are then clipped to X_{max}. The children's velocity vectors are averaged and the previous best vector is set to the new position vector, effectively restarting the children's memory.

The velocity and position update rules remain unchanged from the standard implementation of the PSO. The social parameter is set to 2.0, inertia is linearly decreased from $0.8 \rightarrow 0.2$ and V_{max} is set to X_{max}.

2 Test Problems

Four numeric optimization problems were chosen to compare the relative performance of the Breeding Swarm algorithm to GA and PSO. These functions are standard test functions used in other evolutionary studies. The first two functions are unimodal, while the remaining two are multimodal. All functions are designed to have a global minima near the origin.

The first test function is the Ellipsoidal function given by the equation:

$$f_0(x) = \sum_{i=1}^{n} ix_i^2 \tag{9}$$

x is a real-valued vector if dimension n and x_i is the ith element in the vector. The second function is the Rosenbrock function given by:

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2) + (x_i - 1)^2) \tag{10}$$

The third test function is the generalized Rastrigin function given by the equation:

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{11}$$

The final function is the generalized Griewank function given by:

$$f_3(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{12}$$

These functions have been widely used in a number of different studies involving evolutionary optimization (see [1], and others.)

2.1 Initialization Method

Often in comparative experiments, researchers choose to set the initial population uniformly distributed about the search space and usually symmetric about the origin. Furthermore, many test functions have global optima at the origin, including those used in this experiment. Fogel and Beyer has shown that this method of initialization can give a false impression in relative performance [5]. Given that the location of the global optima is generally not known, they suggest that the initialization range not include the global optima. A detailed description of the initialization parameters for all test function are shown in Table 2.

Table 2. Initialization Ranges for Test Problems

function	Asymmetric Initialization Range
f_0	$(50, 100)$
f_1	$(15, 30)$
f_2	$(2.56, 5.12)$
f_3	$(300, 600)$

Each function is ran for 50 trials. Tests using 10 dimensions were allowed to run for 1000 generations. Tests using 20 dimensions were allowed to run for

1500 generations. Finally tests using 30 dimensions were allowed to run for 2000 generations.

3 Results

Figure 1 show the results of each function in 10 dimensions, using asymmetric initialization. Here the BS algorithm was able to find the exact optima (in every trial) in 8 of the 12 test cases (see Table 3). The GA and PSO algorithms were unable to find the exact optima consistently for any test case.

Table 3 show the mean best fitness value along with the variance for each test case using asymmetric initialization. a t-test performed for each case shows the BS algorithm to be statistically significantly better than the GA and PSO in all test cases. The GA algorithm was able to outperform the PSO algorithm in the unimodal test function f_2

Table 3. Asymmetric Initialization. Summary of mean best results in the final generation for each test. t-scores are all significant.

Problem	Dims	Gens	Real GA Mean Best (variance)	PSO Mean Best (variance)	BS Mean Best (variance)	t-score BS GA	t-score BS PSO
f0	10	1000	11.637 (34.403)	3.577e-043 (1.139e-042)	0 (0)	14.029	15.702
	20	1500	261.971 (11569.1)	3.436e-021 (2.388e-020)	0 (0)	17.222	7.194
	30	2000	2476.92 (954655)	2.812e-010 (1.317e-009)	0 (0)	17.925	10.676
f1	10	1000	493.004 (496721)	22.685 (48.079)	8.801 (0.014)	4.857	14.443
	20	1500	2971.7 (3.194e+006)	43.291 (62.746)	18.915 (0.028)	11.682	19.433
	30	2000	12741 (2.221e+007)	70.114 (76.726)	28.996 (0.003)	19.072	26.796
f2	10	1000	4.272 (4.551)	12.138 (7.021)	0 (0)	14.160	86.441
	20	1500	42.458 (116.344)	60.095 (20.520)	0 (0)	27.834	146.430
	30	2000	122.984 (600.81)	134.538 (35.421)	0 (0)	35.478	189.913
f3	10	1000	0.676 (0.0287)	0.095 (0.038)	0 (0)	28.200	125
	20	1500	1.824 (0.124)	0.057 (0.166)	0 (0)	36.627	17.169
	30	2000	5.729 (1.627)	0.193 (0.333)	0.005 (0.001)	31.711	28.313

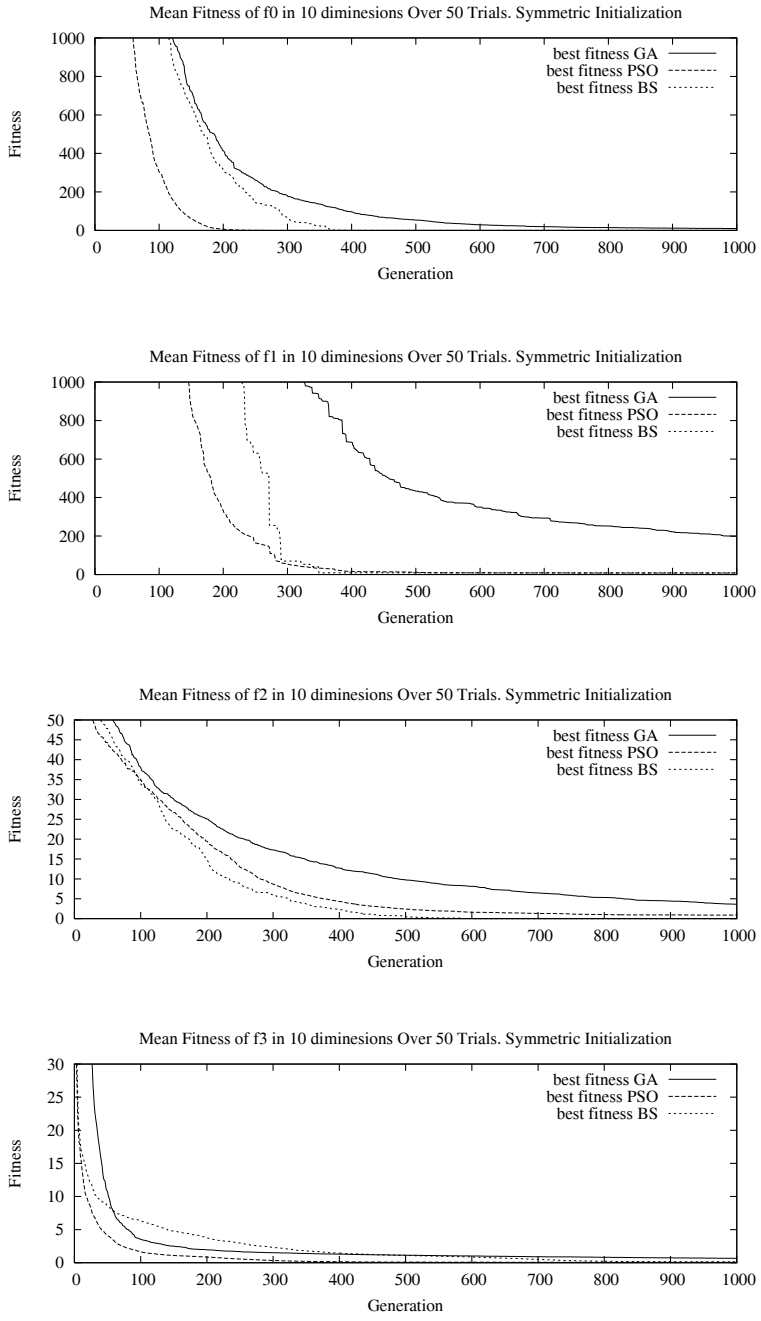

Fig. 1. PSO, GA and BS Results for the test problems over 50 Trials.

In many test cases the PSO algorithm was able to find near optimal solutions in a majority of trials. For the majority of these cases the population would converge on a suboptimal solution preventing the population from improving. BS did not have this problem, the GA inspired VPAC operator prevented the population from converging on a suboptimal solution by consistently pushing members of the population away from the global best particle.

4 Conclusions

As the results above demonstrate, the performance of Breeding Swarms is competitive with both the GA and PSO, in this test performing better than both. The breeding swarm algorithm was able to locate a optimal, or near optimal, solution significantly faster than either GA or PSO. This is most likely due to the VPAC crossover actively dispersing the population, allowing the population to cross greater distances in the search space faster.

The Breading Swarm algorithm developed here is highly customizable in order to allow future researchers large latitude in implementing the algorithm. Future research will include investigation into elitism and its effect on performance. Different selection and crossover types and different values for the breeding ratio parameter and its effects on performance. The use of a mutation operator also needs to be investigated.

An additional advantage of the Breeding Swarm algorithm may be the ability to implement a variable length string to represent potential solutions, a trait not available in the standard PSO implementation. Through the crossover operator, the individual may be allowed to grow or shrink depending on its needs, such as in GAs. Further research will look into this possibility.

Acknowledgment

This work is supported by NSF EPSCoR EPS-0132626. These experiments were performed on a Beowulf cluster built with funds from NSF grant EPS-80935 and a generous hardware donation form Micron Technologies. Thanks to Dr. Hiromoto of the University of Idaho for a conversation that sparks many of these ideas.

References

1. Peter J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V. William Porto and et al., editors, *Evolutionary Programming*, volume 1447 of *Lecture Notes in Computer Science*, pages 601–610. Springer, 1998.
2. Morten Bek, Troels Grosbll-Poulsen, and Mads Ulrik Kristofferson. Evolutionary trained kohonen networks as classifiers for human utterances. In *Master's Thesis*. Department of Computer Science, University of Aarhus, 2002.

3. Russell C. Eberhart and Yuhui Shi. Comparison between genetic algorithms and particle swarm optimization. In et. al. V. William Porto, editor, *Evolutionary Programming*, volume 1447 of *Lecture Notes in Computer Science*, pages 611–616. Springer, 1998.
4. Larry J. Eshelman and J. David Schaffer. Real-coded genetic algorithms and interval-schemata. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.
5. David B. Fogel and Hans-Georg Beyer. A note on the empirical evaluation of intermediate recombination. *Evolutionary Computation*, 3(4):491–495, 1996.
6. J.H. Holland. *Adaptation in natural and artificial systems: 2nd edn.* The University of Michigan Press, Ann Arbor, MI, 1992.
7. J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2001.
8. M. Lvbjerg, T. Rasmussen, and T. Krink. Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2001*, LNCS. Springer-Verlag, 2001.
9. Clerc M. The swarm and the queen: Towards a determininistic and adaptive particle swarm optimization. In *Congress on Evolutionary Computation (CEC99)*, pages 1951–1957, 1999.
10. J. Robinson, S.Sinton, and Y. Rahmat-Samii. Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna. In *IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, San Antonio, TX, 2002.
11. Matthew Settles, Brandon Rodebaugh, and Terence Soule. Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In E. Cantú-Paz and et. al., editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 148–149, Chicago, 12-16 July 2003. Springer-Verlag.
12. Matthew Settles and Terence Soule. A hybrid ga/pso to evolve artificial recurrent neural networks. In C. Dagli and et. al., editors, *Intelligent Engineering Systems Through Aritificial Neural Networks (ANNIE-2003)*, volume 13, pages 51–56, St. Louis, 2-5 Nov. 2003. ASME Press.
13. Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In V. William Porto and et al., editors, *Evolutionary Programming*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer, 1998.