A Diversity-Guided Particle Swarm Optimizer – the ARPSO

Jacques Riget and Jakob S. Vesterstrøm
EVALife Project Group
Department of Computer Science
Aarhus Universitet, Bgn. 540, Ny Munkegade
DK-8000 Aarhus C, Denmark
Phone: +45 89423371
riget@daimi.au.dk, jve@daimi.au.dk, www.evalife.dk

EVALife Technical Report no. 2002-02

Abstract

The particle swarm optimization (PSO) algorithm is a new population based search strategy, which has exhibited good performance on well-known numerical test problems. However, on strongly multi-modal test problems the PSO tends to suffer from premature convergence. This is due to a decrease of diversity in search space that leads to a total implosion and ultimately fitness stagnation of the swarm. An accepted hypothesis is that maintenance of high diversity is crucial for preventing premature convergence in multi-modal optimization.

We introduce the *attractive* and *repulsive* PSO (ARPSO) in trying to overcome the problem of premature convergence. It uses a diversity measure to control the swarm. The result is an algorithm that alternates between phases of *attraction* and *repulsion*. The performance of the ARPSO is compared to a basic PSO (bPSO) and a genetic algorithm (GA). The results show that the ARPSO prevents premature convergence to a high degree, but still keeps a rapid convergence like the basic PSO. Thus, it clearly outperforms the basic PSO as well as the implemented GA in multi-modal optimization.

Keywords

Particle Swarm Optimization, Diversity-Guided Search

1 Introduction

The PSO model is a new population based optimization strategy introduced by J. Kennedy et al. in 1995 (Kennedy95). It has already shown to be comparable in performance with traditional optimization algorithms such as simulated annealing (SA) and the genetic algorithm (GA) (Angeline98; Eberhart98; Krink01; Vesterstrom01).

A major problem with evolutionary algorithms (EAs) in multi-modal optimization is premature convergence (PC), which results in great performance loss and sub-optimal solutions. As far as GAs are concerned, the main reason for premature convergence is a too high selection pressure or a too high gene flow between population individuals. With PSOs the fast information flow between particles seems to be the reason for clustering of particles. Diversity declines rapidly, leaving the PSO algorithm with great difficulties of escaping local optima. Consequently, the clustering leads to low diversity with a fitness stagnation as an overall result.

The problem with premature convergence will always persist, since we obviously must check the whole search-space in order to ensure that a result is not sub-optimal. In spite of this fact, and although the goals of maintaining high diversity and obtaining fast convergence is partially contradicting, it makes perfectly good sense to try to improve the optimization algorithm, in order to avoid sub-optimal solutions more frequently. Several papers have been written on this subject during the last three decades. Many solutions and improvements have been suggested, especially for the GA (Bäck97; DeJong75; Goldberg87; Krink00; Thomsen00; Ursem99).

With respect to the PSO model only few papers have been written on the subject; and even fewer have accomplished the goal of dealing with premature convergence. Proposed solutions to the problem include the addition of a queen particle, the alternation of the neighborhood topology, the introduction of subpopulations and giving the particles a physical extension (Clerc98; Kennedy99; Lovbjerg01; Krink01). Based upon the few results for the PSO model concerning premature convergence we present, in this paper, a new model which deals with this issue. The modified PSO model is especially designed for multi-modal optimization. It is indeed an effective model, yet conceptually it is simple as well as being very easy to implement.

Recently R. Ursem has suggested a model called the Diversity-Guided Evolutionary Algorithm (DGEA) (Ursem01). He redefines the traditional mutation operator, the Gaussian mutation, to be a directed mutation instead. The important issue is that this directed mutation, in general, increases the diversity, whereas normal Gaussian mutation is not likely to do this, because it simply adds random noise from some distribution with a mean of zero, normally $N(0, \sigma^2)$. Consequently, the DGEA applies diversity-decreasing operators (selection, recombination) and diversity-increasing operators (mutation) to alternate between two modes based upon a distance-to-average-point measure. The performance of the DGEA clearly shows its potential in multi-modal optimization.

As Ursem rightfully pinpoints, the diversity measure is traditionally used to *analyze* the evolutionary algorithms rather than *guide* them. We are great believers of *adaptive controlling*; that measuring and using different properties of the swarm/population while running, adds significant potential to the algorithm. We have therefore adopted the idea from Ursem with the decreasing and increasing diversity operators used to control the population into the basic PSO model. We find, it is a natural modification of the PSO, and the idea behind it is surprisingly simple. The modified model uses a diversity measure to have the algorithm alternate between exploring and exploiting behavior. We introduce two phases: *attraction* and *repulsion.* By measuring the diversity we let the swarm alternate between these phases. As long as the diversity is above a certain threshold d_{low} the particles attract each other. When the diversity declines below d_{low} the particles simply switch and start to repel each other until the threshold d_{high} is met. With this simple scheme we obtain our modified model, which we have chosen to call the ARPSO model – the attractive and repulsive PSO.

2 The PSO Models

2.1 The Basic PSO Model

The basic PSO model consists of a swarm of particles moving in an n-dimensional, real-valued search space of possible problem solutions. For the search space, in general, a certain quality measure, the fitness, is defined making it possible for particles to compare different problem solutions. Every particle has a position vector $\vec{x}$ and a velocity vector $\vec{v}$. Moreover, each particle contains a small memory storing its own best position seen so far $\vec{p}$ and a global best position $\vec{g}$ obtained through communication with its fellow neighbor particles. This information flow is obtained by defining a neighborhood topology on the swarm telling particles about immediate neighbors.[1]

The intuition behind the PSO model is that by letting information about good solutions

[1]For more information on neighborhood topology we refer to (Kennedy99) and (Krink01).

```
Program PSO
  init();
  while not done do
    setDirection();              // new!
    updateVelocity();
    newPosition();
    assignFitness();
    calculateDiversity();        // new!
```

Figure 1: The ARPSO algorithm.

spread out through the swarm, the particles will tend to move to good areas in the search space. At each time step t the velocity is updated and the particle is moved to a new position. This new position is simply calculated as the sum of the previous position and the new velocity:

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1). \tag{1}$$

The update of the velocity from the previous velocity to the new velocity is, as implemented in this paper, determined by:

$$\vec{v}(t+1) = \omega \cdot \vec{v}(t) + \phi_1(\vec{p}(t) - \vec{x}(t)) + \phi_2(\vec{g}(t) - \vec{x}(t)), \tag{2}$$

where ϕ_1 and ϕ_2 are real numbers chosen uniformly and at random in a given interval, usually $[0, 2]$. These values determine the significance of $\vec{p}(t)$ and $\vec{g}(t)$ respectively. The parameter ω is the inertia weight and controls the magnitude of the old velocity $\vec{v}(t)$ in the calculation of the new velocity $\vec{v}(t+1)$.[2]

2.2 The Modified Model - ARPSO

We define the attraction phase merely as the basic PSO algorithm. The particles will then attract each other, since in general they attract each other in the basic PSO algorithm because of the information flow of good solutions between particles. We define the second phase repulsion, by "inverting" the velocity-update formula of the particles:

$$\vec{v}(t+1) = \omega \cdot \vec{v}(t) - \phi_1(\vec{p}(t) - \vec{x}(t)) - \phi_2(\vec{g}(t) - \vec{x}(t)). \tag{3}$$

In the repulsion phase the individual particle is no longer attracted to, but instead repelled by the best known particle position $\vec{g}(t)$ and its own previous best position $\vec{p}(t)$.

In the attraction phase the swarm is contracting, and consequently the diversity decreases. When the diversity drops below a lower bound, d_{low}, we switch to the repulsion phase, in which the swarm expands due to the above inverted update-velocity formula 3. Finally, when a diversity of d_{high} is reached, we switch back to the attraction phase. The result of this is an algorithm that alternates between phases of exploiting and exploring – attraction and repulsion – low diversity and high diversity. The pseudo-code for the ARPSO algorithm is shown in figures 1 and 2.

```
Function setDirection
  if (dir > 0 && diversity < dLow) dir = -1;
  if (dir < 0 && diversity > dHigh) dir = 1;
```

Figure 2: setDirection

The first of the two new functions, `setDirection` determines which phase the algorithm is currently in, simply by setting a sign-variable, *dir*, either to 1 or -1 depending on the

[2]For a more generalized velocity update formula we refer to (Kennedy99).

diversity. In the second function, `calculateDiversity`, the diversity of the swarm (in the pseudo-code stored in the variable "diversity"), is set according to the diversity-measure:

$$diversity(S) = \frac{1}{|S| \cdot |L|} \cdot \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^{N} (p_{ij} - \overline{p}_j)^2}, \tag{4}$$

where S is the swarm, $|S|$ is the swarmsize, $|L|$ is the length of longest the diagonal in the search space, N is the dimensionality of the problem, p_{ij} is the j'th value of the i'th particle and $\overline{p}_j$ is the j'th value of the average point $\overline{p}$. Note that this diversity measure is independent of swarmsize, the dimensionality of the problem as well as the search range in each dimension.

Finally, the velocity-update formula, eq. 2, is changed by multiplying the sign-variable *dir* to the two last terms in it. This decides directly whether the particles attract or repel each other:

$$\vec{v}(t+1) = \omega \cdot \vec{v}(t) + dir(\phi_1(\vec{p}(t) - \vec{x}(t)) + \phi_2(\vec{g}(t) - \vec{x}(t))). \tag{5}$$

3 Experimental Settings

3.1 Benchmark Functions

We have tested the modified PSO model on four standard multi-modal objective functions. All four are widely known benchmark functions for testing the performance of different evolutionary optimization strategies such as evolutionary programming, simulated annealing, genetic algorithms and particle swarm optimization. The four test functions are:

Griewank, n-dimensional

$$f_1(\vec{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{6}$$

$$where - 600 \leq x_i \leq 600$$

AckeyF1, n-dimensional

$$f_2(\vec{x}) = e + 20 - 20 \cdot exp\Big(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\Big) - exp\Big(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi \cdot x_i)\Big), \tag{7}$$

$$where - 30 \leq x_i \leq 30$$

Rosenbrock, n-dimensional

$$f_3(\vec{x}) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{8}$$

$$where - 100 \leq x_i \leq 100$$

Rastrigin, n-dimensional

$$f_4(\vec{x}) = \sum_{i=1}^{n} x_i^2 + 10 - 10 \cdot \cos(2\pi x_i) \tag{9}$$

$$where - 5.12 \leq x_i \leq 5.12$$

In our selection, several aspects of multi-modal optimization are represented.[3] Each test function has been chosen to have unique characteristics for the purpose of extracting as much information about our modified PSO model as possible. The Griewank test function is Sphere-like with added noise (the $\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$ term). In low dimensions this is a highly multi-modal function, whereas in higher dimensions the Griewank function resembles the plain Sphere-function, because the added noise diminishes ($\lim_{n\to\infty}\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) = 0$ for randomly chosen $\vec{x}$). The Rosenbrock function is extremely steep when the optimum is being approached from far and "banana"-shaped close to the optimum. Additionally, the Rosenbrock function, as the only one, has a strong dependency between variables x_i. The AckeyF1 and Rastrigin objective functions are both highly multi-modal in all dimensions, but they have different steepness. All test functions have a global minimum at $(0, 0, \ldots, 0)$ with a fitness value of 0.

3.2 Settings for the PSO Models

The basic PSO and the ARPSO model were implemented as described in section 2. Based upon the findings of Krink et al. (Krink01), we applied a fully connected neighborhood topology, meaning that every particle memory $\vec{g}(t)$ is updated with the global best known position in each timestep (Krink01). Problem dependent parameters such as the *swarmsize*, v_{max} as well as the inertia weight ω, were tuned for each test case. Furthermore, we optimized the ω to be a linearly decreasing function $l(t) = 1 - \frac{t}{t_{max}}$ of time t to emphasize local search towards the end of the test run.[4] The diversity parameters d_{low} and d_{high} were set at $5.0 \cdot 10^{-6}$ and 0.25 respectively.[5]

3.3 Genetic Algorithm

The genetic algorithm used in the comparison is more or less a straight-forward implementation of the standard, real-valued GA. It uses tournament-selection with a tournamentsize of 2, one-point crossover (applied to an individual with probability p_c) and a N(0,σ^2)-distributed mutation operator (applied independently at each gene-coordinate with probability p_m). We optimized the variance parameter σ^2 to be a decreasing function of time to emphasize local search towards the end of the test run. The variance was either set to the linearly decreasing $\sigma^2(t) = 1 - \frac{t}{t_{max}}$ or the hyperbolically decreasing $\sigma^2(t) = \frac{1}{1+\sqrt{t}}$ of time t. Furthermore we used elitism with a size of 1. All problem dependent parameters p_c, p_m, the variance σ^2 as well as populationsize were optimized for each test case.

3.4 Experiments

Two different sets of experiments were conducted: 1) The standard comparison of performance between the different algorithms, and 2) Measurement of the percentage of time spent and the percentage of improvements obtained in the two phases attraction and repulsion in the ARPSO.

We made experiments with each test function in 20, 50, and 100 dimensions. The performance of the modified model is directly compared to the basic PSO model and the implemented GA. The number of evaluations was set to 2000 times the dimensionality of the test function.[6] In preliminary tests the ARPSO continued to improve the fitness at the end of the fixed number of evaluations. Hence, better fitness values could be obtained if the ARPSO ran until stagnation, which we defined to be 200000 evaluations without fitness

[3]see appendix D for further information on benchmark functions.

[4]t_{max} denotes the time when the algorithm stops.

[5]These settings are copied form [Ursem01] and proved to be reasonable in preliminary tests.

[6]i.e the number of evaluations in the different dimensions is 20d = 400000 evaluations, 50d = 1000000 evaluations, and 100d = 2000000 evaluations.

improvement. This optimized ARPSO model is denoted ARPSO*. All results presented in this paper are averages of 50 repeated runs.

4 Experimental Results

Table 8.1 shows the results from the conducted experiments. Figures 3 to 6 show the performance over time for some of our experiments: They all have average best fitness plotted for each evaluation for the implemented GA, the basic PSO, and the diversity controlled ARPSO. Furthermore the results from the second experiment of time/improvements in the two phases are shown in table 8.2.

Dim.	**20**	**50**	**100**
Performance on Griewank			
GA	$1.71 \cdot 10^{-2}$	$8.51 \cdot 10^{-2}$	$2.39 \cdot 10^{-1}$
bPSO	$1.74 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$	$1.25 \cdot 10^{-2}$
ARPSO	$2.50 \cdot 10^{-2}$	$3.05 \cdot 10^{-2}$	$9.84 \cdot 10^{-2}$
ARPSO*	$2.40 \cdot 10^{-2}$	$2.99 \cdot 10^{-2}$	$3.74 \cdot 10^{-2}$
Performance on AckeyF1			
GA	0.012	0.376	0.389
bPSO	0.018	0.668	0.830
ARPSO	$0.33 \cdot 10^{-7}$	0.027	0.218
ARPSO*	$0.30 \cdot 10^{-7}$	$0.96 \cdot 10^{-2}$	0.015
Performance on Rosenbrock			
GA	107.10	199.41	254.00
bPSO	11.16	30.08	122.14
ARPSO	2.34	10.43	103.46
ARPSO*	$1.67 \cdot 10^{-3}$	0.116	88.71
Performance on Rastrigin			
GA	14.43	45.36	63.07
bPSO	9.71	47.14	96.59
ARPSO	**0**	$0.20 \cdot 10^{-1}$	0.438
ARPSO*	**0**	**0**	**0**

Table 1: Average best fitness for the implemented GA, the basic PSO and the ARPSO on the four benchmark problems. The precision is down to $1 \cdot 10^{-10}$. Test runs that reached below $0.5 \cdot 10^{-10}$ are denoted with a **0**. The results in the rows ARPSO* were obtained by running the ARPSO until fitness stagnation in 200000 evaluations.

The results on performance clearly show that the ARPSO is a much stronger optimizer than the basic PSO and the implemented GA on all the test problems except on the Griewank function. More precisely, stronger means that the ARPSO's best fitness value obtained is several magnitudes better than that of the basic PSO and GA. Additionally, the ARPSO has a better or at least as good convergence speed as the basic PSO algorithm.

On the Griewank benchmark function, the ARPSO is not performing better than the basic PSO and it is obvious why: Griewank is essentially unimodal when the dimensionality exceeds 10-15 as explained is section 3.1. Hence the basic PSO has no significant difficulties in finding the global optimum. When the ARPSO is tested on Griewank function, it uses approximately 20 % of its evaluations in the repulsion phase[7], but these are more or less "wasted", because the basic PSO easily finds a near optimal solution without repulsion. Moreover, the ARPSO has reduced possibilities of fine-tuning the found optimum due to the

[7]see table 8.2 on the Griewank function.

	Time spent in		Improvements in	
Problem	Attraction	Repulsion	Attraction	Repulsion
Ackey20	76.9 %	23.1 %	98.5 %	1.5 %
Ackey50	78.4 %	21.6 %	98.9 %	1.1 %
Ackey100	81.1 %	18.9 %	98.9 %	1.1 %
Griewank20	76.8 %	23.2 %	98.2 %	1.8 %
Griewank50	78.6 %	21.4 %	97.1 %	2.9 %
Griewank100	80.1 %	19.9 %	96.4 %	3.6 %
Rastrigin20	83.6 %	16.4 %	98.9 %	1.1 %
Rastrigin50	85.6 %	14.4 %	98.8 %	1.2 %
Rastrigin100	88.6 %	11.4 %	98.9 %	1.1 %
Rosenbrock20	69.2 %	30.8 %	88.8 %	11.2 %
Rosenbrock50	71.3 %	28.7 %	89.3 %	10.7 %
Rosenbrock100	79.2 %	20.8 %	95.4 %	4.6 %

Table 2: Percentage of time spent and percentage of improvements made in the attraction- and repulsion-phase by the ARPSO algorithm.

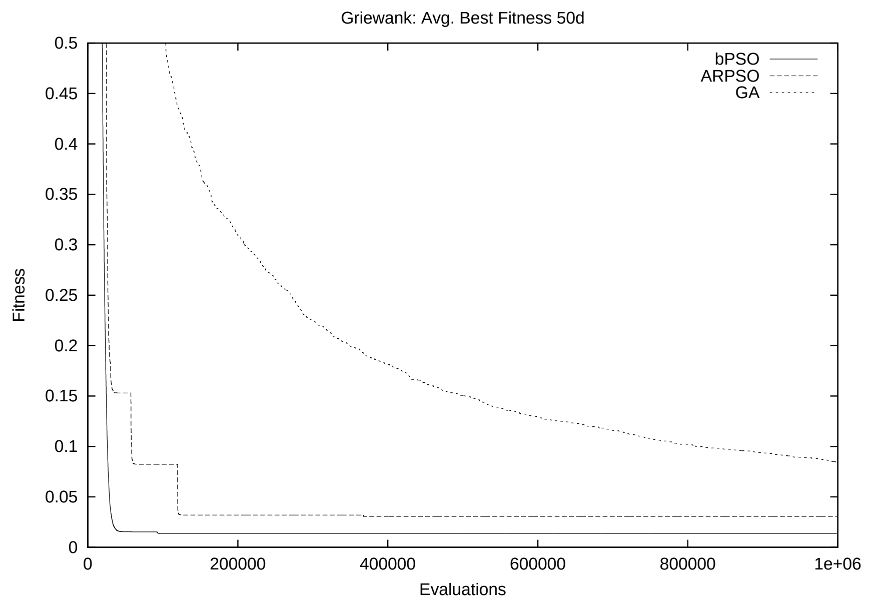

Figure 3: Performance on Griewank, 50 dimensions

fact that it constantly alternates between the two phases. Consequently, the ARPSO shows a marginally worse performance than the basic PSO on the Griewank function.

However, on the Ackey, the Rosenbrock, and the Rastrigin test problems, the basic PSO algorithm cannot easily find the global optimum. Often it is never found. On these problems the ARPSO is without any exception a great improvement as it can be seen directly on the graphs 4 to 6 as well as in table 8.1. The ARPSO algorithm simply finds the global optimum more often, and the obtained fitness value is on average much better than that of the basic PSO. The greatest improvements are found for the Ackey and the Rastrigin benchmark functions.

When the ARPSO is run until no further improvement occurs, we ascertain that the algorithm is able to improve further and reach an even better fitness value – it does not

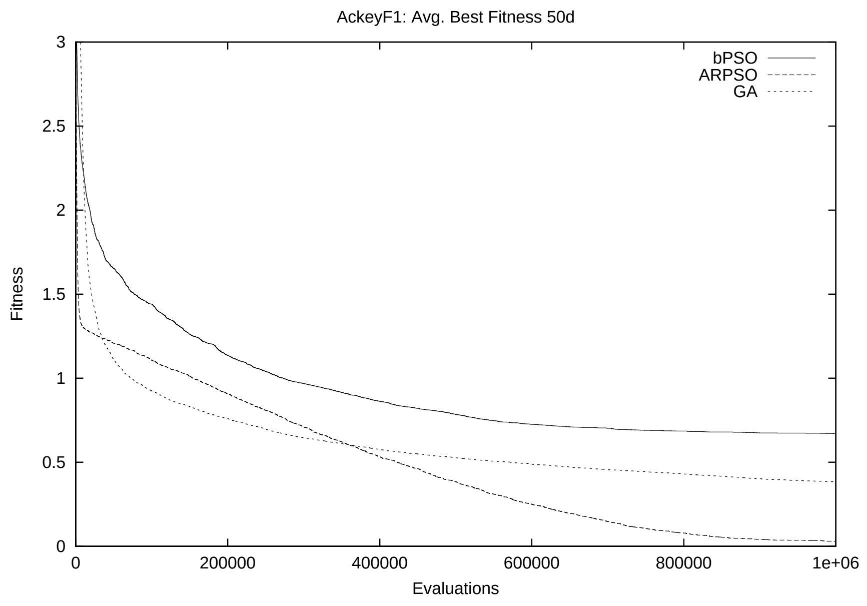

Figure 4: Performance on AckeyF1, 50 dimensions

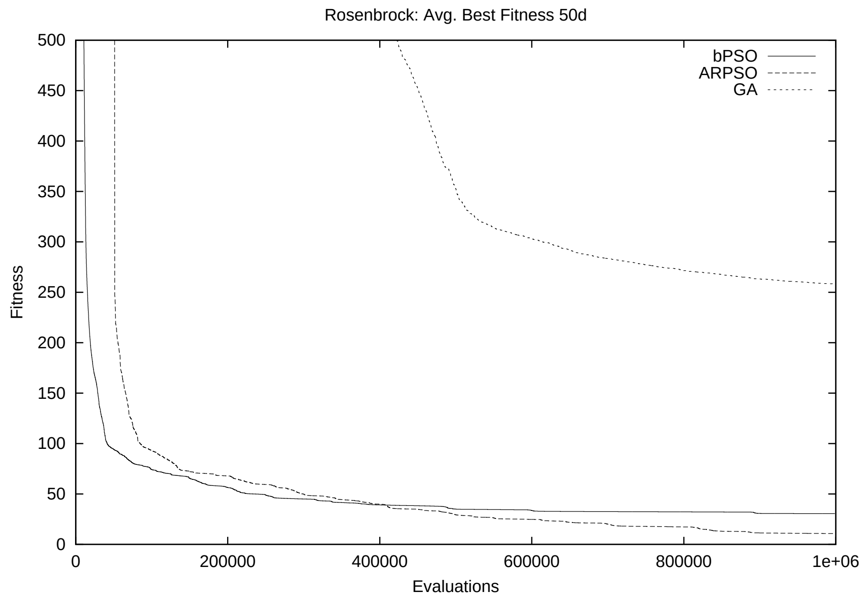

Figure 5: Performance on Rosenbrock, 50 dimensions

stagnate until after several million evaluations; good examples are Rosenbrock and Rastrigin in 50 dimensions.

5 Discussion

5.1 Performance of the ARPSO

The motivation for our approach of controlling the diversity is to overcome the premature convergence that the basic PSO model suffers from. In this paper we have introduced the novel ARPSO model, which is a simple extension of the basic PSO with two phases: *attraction*

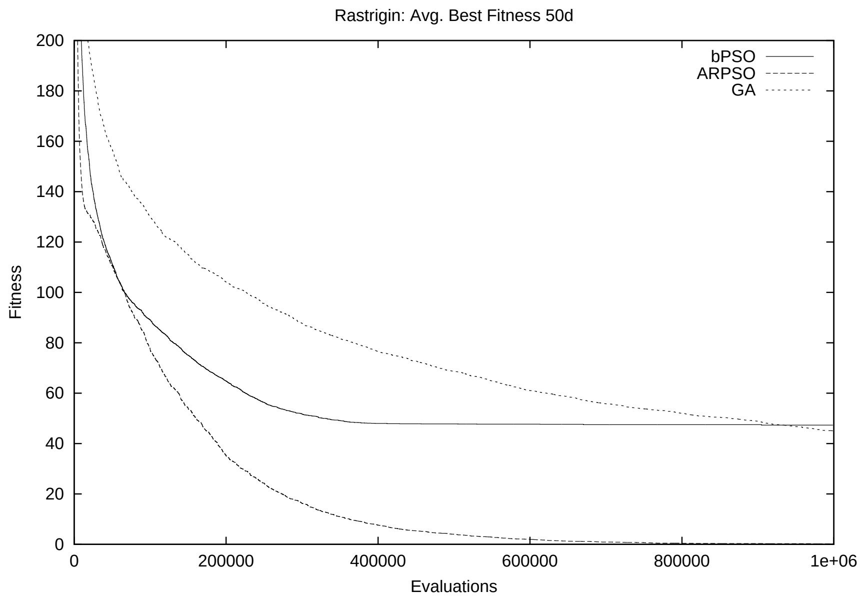

Figure 6: Performance on Rastrigin, 50 dimensions

and *repulsion*. Our results on performance listed in table 8.1 and shown in figures 3 to 6 clearly show that the ARPSO outperforms the other algorithms by several magnitudes on the multi-modal benchmark problems except for the Griewank function. We have explained this by the diminishment of local optima, which essentially makes Griewank a uni-modal problem in higher dimensions.

Both the implemented GA and the basic PSO algorithms tend to stagnate prematurely. With genetic algorithms, especially with tournament-selection, a population of many clones have great risk of getting stuck on a local optimum. The diversity in the search space decreases, and the many clones make it difficult for the genetic algorithm to escape a found local optimum. In the basic PSO model the high level of attraction between particles likewise lowers the diversity in such a degree that particles often get trapped on a local optimum.

The ARPSO model does not suffer from this because we control the diversity directly with the parameters d_{low} and d_{high}. Conclusively, the ARPSO algorithm is able to escape local optima, because the repulsion phase heightens the search space diversity, disregarding the particles' fitness values.

5.2 Controlling the Diversity

Low diversity is often blamed for being the main reason for premature convergence. On the other hand low diversity often increases the probability of improving the best fitness value obtained, because the particles will be in the locality of a optimum with chances of fine-tuning solutions. Table 2 shows the percentage of time spent and improvements made by the ARPSO algorithm in the two phases: attraction and repulsion. There is one major point is to be made from this: Almost all fitness improvements occur in the attraction phase even compared to the amount of time the particles spend in this phase. This truly shows the importance of low diversity. Conclusively, the repulsion phase is more or less solemnly used as a diversity-increasing method. Although not thoroughly tested, we have no reason to believe that the improvements in the repulsion phase are of greater importance than the improvements in the attraction phase. This suggests a modification of the ARPSO algorithm, where evaluation is simply omitted in the repulsion phase. Doing so might save a considerable amount of fitness evaluations in the repulsion phase without great loss in the rate of fitness improvement.

Figure 7 compares the diversity between the basic PSO and our ARPSO algorithm on the Rastrigin benchmark function in 50 dimensions. The graph for the basic PSO is an average of 50 test runs. To show the clear diversity changes by means of the ARPSO we present the graph from a single test run. Nonetheless the difference is evident. With the basic PSO the diversity drops to an average level of about $3.0 \cdot 10^{-4}$ and stagnates resulting in the premature convergence and low possibility of significant fitness improvements. In contrast, with the ARPSO we see the diversity varies from less than $d_{low} = 5.0 \cdot 10^{-6}$ to more than $d_{high} = 0.25$. The resulting graph shows enormous peaks right about where the algorithm alternates between the attraction-phase and the repulsion-phase.

In conclusion, there is a great need of high diversity throughout the optimization to prevent premature convergence and stagnation on sub-optimal solutions. Nevertheless the whole optimization process depends on the fitness improvements obtained by low diversity in search space. It is exactly the low diversity, which provides fine-tuning of solutions.

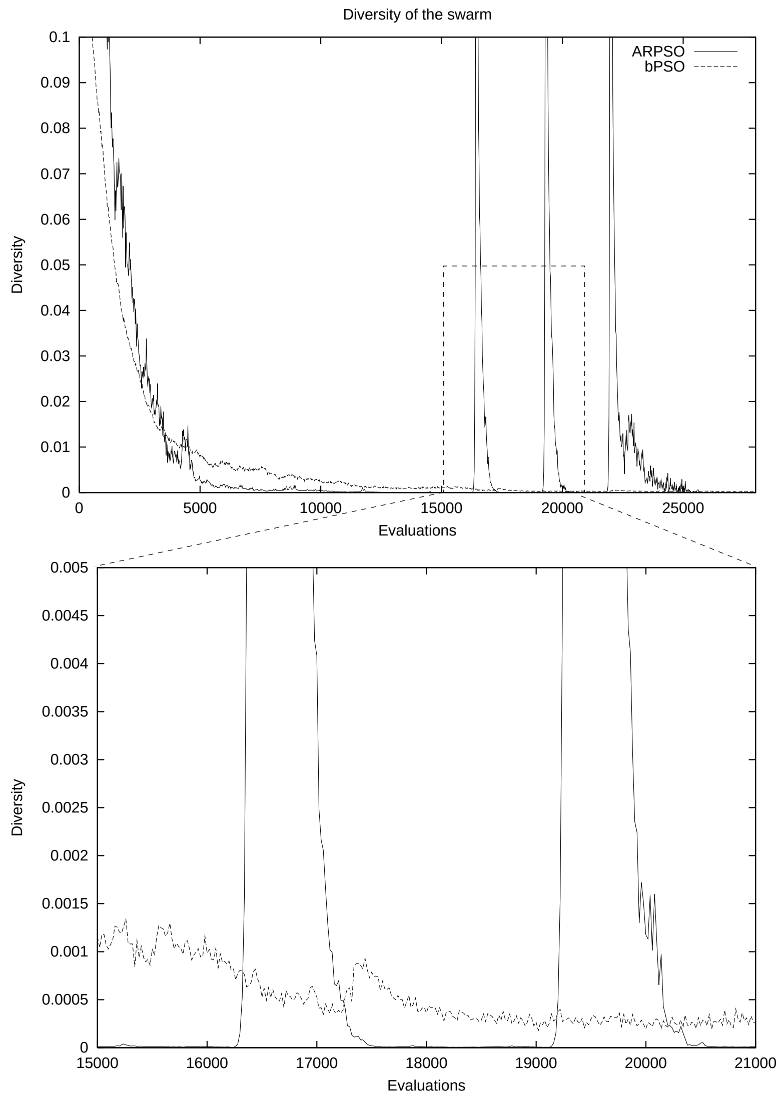

Figure 7: Diversity comparison between the basic PSO and the ARPSO algorithm on the Rastrigin benchmark function in 50 dimensions. The graph for the basic PSO is an average of 50 test runs, whereas the graph for the ARPSO is based upon a single test run.

6 Conclusion

First, we observe that the basic PSO performs approximately as well as the GA on most test problems – on Ackey it is marginally worse, but on Rosenbrock it is far better.

Next, we conclude that the ARPSO on all four test problems performs extremely well. On the essential uni-modal test problem Griewank, the ARPSO and the basic PSO performs equally well. On the truly multi-modal test problems, the ARPSO algorithm by far outperforms the basic PSO and the implemented GA. In fact, it is able to escape the local optima, on which the basic PSO stagnates. Thus, it reaches much better fitness values than the basic PSO. Often the global optimum is found. The ARPSO algorithm to a large extend prevents premature convergence; the short periods with high diversity makes it possible for the algorithm to find new and better areas of the search space. The ARPSO continues to improve for a remarkably long period of time. This is achieved virtually without losing convergence speed at the beginning of the run. Although, as mentioned in section 1, the goals of obtaining fast convergence as well as fine-tuning while maintaining high diversity in the search space are conflicting, we believe the ARPSO model is a strong optimizer with regard to multi-modal optimization. The trade-off cost "paid" regarding convergence rate in return for avoiding premature convergence is minimal to non-existent.

7 Future Work

We have been considering how to improve the ARPSO even further. At least two possibilities could be interesting to investigate into detail.

The first possibility concerns the global best known position $\vec{g}(t)$. In the current version of the ARPSO, this position is kept unchanged in the particles' memory both when entering the attraction- and repulsion-phase. However, it would make a lot of sense to clear $\vec{g}(t)$; for instance when entering the attraction-phase. The whole idea of increasing the diversity of the swarm and then having the particles contract again is to uncover new and better areas of the search space. One could easily imagine that this task would be solved better, if $\vec{g}(t)$ was cleared when entering the attraction-phase, hence allowing the swarm to explore freely in the search space instead of constantly being dragged towards the old $\vec{g}(t)$ during exploration (as it is the case now). We have conducted some preliminary tests on this issue, but these have not shown any decisive conclusion yet.

The second possibility deals with the evaluation of the particles' fitness. As shown in table 8.2, between 11% and 30% of the evaluations are carried out in the repulsion-phase. However, only 1% to 3% of the improvements occur here (with Rosenbrock as an exception). In general, there is a difference of 20%-points between the time spent in the repulsion-phase and the percentage of improvements obtained in it. This suggests that we could omit evaluating the fitness function in the repulsion-phase and consequently save approximately 20% of the total amount of evaluations. Since fitness function evaluation is the dominating factor in the total computation-time of many real-world applications of evolutionary computation, this could be an important point to notice. We have performed some few promising tests to pursue this idea further, and to a large extend these support the above considerations. Figure 8 and 9 clearly show a performance improvement when not evaluating in the repulsion-phase compared to the ARPSO with evaluations in both phases. The ARPSO without evaluations in the repulsion phase uses about 10% to 20% fewer evaluations to reach a certain fitness-value.

References

[Angeline98] P. J. Angeline, "Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 601–610. Springer.

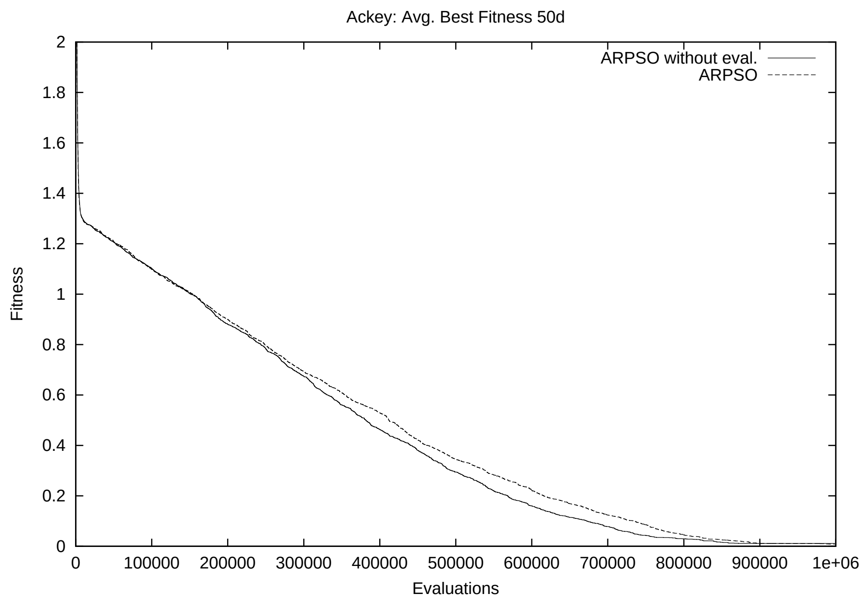

Figure 8: Performance comparison between the basic ARPSO and the ARPSO without evaluation in repulsion-mode on Ackey 50d.

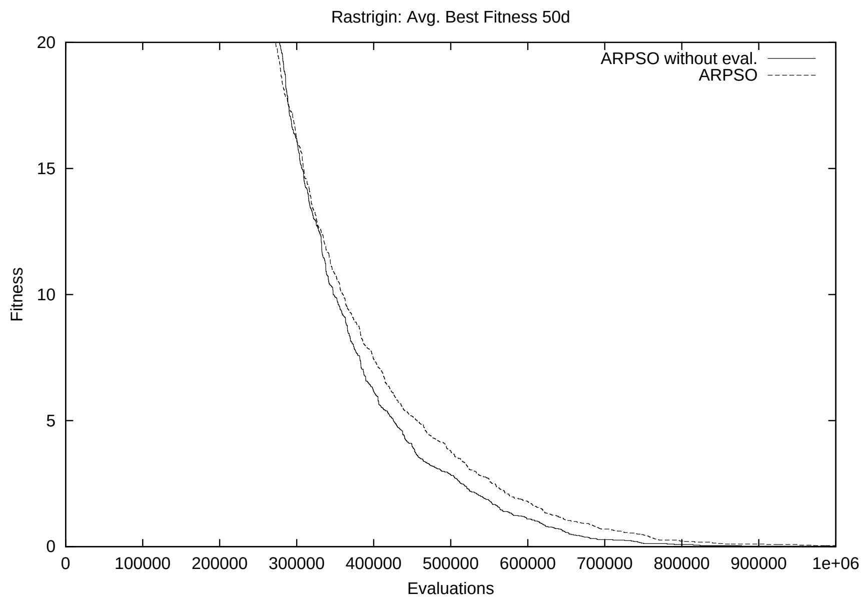

Figure 9: Performance comparison between the basic ARPSO and the ARPSO without evaluation in repulsion-mode on Rastrigin 50d.

[Bäck97] Bäck et al., Handbook on Evolutionary Computation. IOP Publishing Ltd. and Oxford University Press. Chapter 6.3 and 6.4

[Clerc98] M. Clerc, The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization

[DeJong75] DeJong, K. A., ”An analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD Thesis, University of Michigan, 1975.

[Eberhart98] R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 611–616. Springer.

[Goldberg87] Goldberg, D. E. and Richardson, J., "Genetic Algorithms with Sharing for Multi-modal Function Optimization", In Grefenstette, editor, "Genetic Algorithms and their Applications" (ICGA'87), pages 41-49.

[Kennedy95] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, 1942–1948. IEEE Press.

[Kennedy99] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931–1938. IEEE Press.

[Lovbjerg01] Løvbjerg, M., Rasmussen, T., K. and Krink, T. "Hybrid Particle Swarm Optimiser with Breeding and Subpopulations". Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001).

[Krink00] T. Krink, R. Thomsen, and P. Rickers. "Applying Self-Organised Criticality to Evolutionary Algorithms". In Parallel Problem Solving form Nature - PPSN VI, volume 1, 375–384.

[Krink01] T. Krink, J. Vesterstrøm, J. Riget. "Particle Swarm Optimization with Spatial Particle Extension". To appear in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC-2002).

[Shi98] Y. Shi and R. C. Eberhart, "Parameter Selection in Particle Swarm Optimization", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 591–600. Springer.

[Shi98B] Y. Shi and R. C. Eberhart, "A modified Particle Swarm Optimiser", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998.

[Shi99] Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1945–1950. IEEE Press.

[Thomsen00] R. Thomsen, P. Rickers., and T. Krink. "A Religion-Based Spatial Model For Evolutionary Algorithms". In Parallel Problem Solving form Nature - PPSN VI, volume 1, 817–826.

[Ursem99] R. K. Ursem, "Multinational Evolutionary Algorithms", Proceedings of the 1999 Congress of Evolutionary Computation (CEC-1999) , vol. 3, 1633–1640. IEEE Press.

[Ursem01] R. K. Ursem, "Diversity-Guided Evolutionary Algorithms", In submission for the Parallel Problem Solving form Nature Conference (PPSN VII).

[Vesterstrom01] J. Vesterstrøm, J. Riget and T. Krink. Division of Labor in Particle Swarm Optimization. To appear in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC-2002).