Fitness-Distance-Ratio Based Particle Swarm Optimization

Thanmaya Peram, Kalyan Veeramachaneni and Chilukuri K. Mohan*
2-177 CST, Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244-4100,USA
tperam, kveerama, mohan@ecs.syr.edu

Abstract - This paper presents a modification of the particle swarm optimization algorithm (PSO) intended to combat the problem of premature convergence observed in many applications of PSO. The proposed new algorithm moves particles towards nearby particles of higher fitness, instead of attracting each particle towards just the best position discovered so far by any particle. This is accomplished by using the ratio of the relative fitness and the distance of other particles to determine the direction in which each component of the particle position needs to be changed. The resulting algorithm (FDR-PSO) is shown to perform significantly better than the original PSO algorithm and some of its variants, on many different benchmark optimization problems. Empirical examination of the evolution of the particles demonstrates that the convergence of the algorithm does not occur at an early phase of particle evolution, unlike PSO. Avoiding premature convergence allows FDR-PSO to continue search for global optima in difficult multimodal optimization problems.

I. INTRODUCTION

The Particle Swarm Optimization algorithm (PSO), originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart in 1995, has come to be widely used as a problem-solving method in engineering and computer science. PSO was inspired from studies of various animal groups, and has since proven to be a powerful competitor to other evolutionary algorithms such as genetic algorithms [3]. Several researchers have analyzed the performance of the PSO and its variants with different settings, e.g., neighborhood settings [5], hybrid PSO with breeding and subpopulations [6]. Work presented in [7] describes the complex task of parameter selection in a PSO model. Comparisons between PSO and the standard GA were done analytically and also with regards to performance in [3].

The PSO algorithm simulates social behavior among individuals (particles) “flying” through a multidimensional search space, each particle representing a point at the intersection of all search dimensions. The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their “best” positions; then use those memories to adjust their own velocities, and thus positions. The original PSO formulae developed by Kennedy and Eberhart[1][2] were modified by Shi and Eberhart [4] with the introduction of an inertia parameter, $\omega$, that was shown empirically to improve the overall performance of PSO.

The PSO formulae define each particle as a potential solution to a problem in a D-dimensional space[8], with the $i$th particle represented as $X_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{iD})$. Each particle also maintains a memory (pbest) of its previous best position, $P_i = (p_{i1}, p_{i2}, p_{i3}, ..., p_{iD})$ and a velocity along each dimension represented as $V_i = (v_{i1}, v_{i2}, v_{i3}, ..., v_{iD})$ [8]. In each generation, the pbest vector of the particle with the best fitness in the local neighborhood, designated gbest, and the pbest vector of the current particle are combined to adjust the velocity along each dimension; the velocity is then used to compute a new position for the particle. The portion of the adjustment to the velocity influenced by the individual’s own pbest position is considered as the cognition component, and the portion influenced by gbest is the social component.

In early versions of PSO, the update formulae were:

$$V_{id}^{(t+1)} = \omega \times V_{id}^{(t)} + \psi_1 \times (p_{id} - X_{id}) + \psi_2 \times (g_{id} - X_{id})$$

(1)

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)}$$

(2)

Constants $\psi_1$ and $\psi_2$ determine the relative influence of the social and the cognition components, and are often both set to the same value to give each component (the cognition and the social learning rates) equal weight. A constant,
This paper proposes a significant modification to the dynamics of particles in PSO, moving each particle towards other nearby particles that have better fitness, instead of just the best position discovered so far. This is in addition to the terms in the original PSO update equations, in which it learns from its own experience and also from the best particle’s experience.

The paper describes the modified particle swarm optimization (FDR-PSO) algorithm and compares it with the standard PSO. The FDR-PSO algorithm is applied to several benchmark continuous optimization problems, and the results illustrate the significant performance improvement achieved over the standard PSO.

Section II motivates and describes the FDR-PSO algorithm and gives the pseudocode for the algorithm. Section III defines the benchmark problems used for experimental comparison of the algorithms, and the experimental settings for each algorithm. Section IV presents the results. Finally the conclusions and future work are presented, with a discussion of related work.

II. FDR-PSO ALGORITHM

Theoretical results [13][14] have shown that the particle positions in PSO oscillate in damped sinusoidal waves until they converge to points in between their previous best positions and the global best positions discovered by all particles so far. If some point visited by a particle during this oscillation has better fitness than its previous best position (which is very likely to happen in many fitness landscapes), then particle movement continues, generally converging to the global best position discovered so far. All particles follow the same behavior, quickly converging to a good local optimum of the problem. However, if the global optimum for the problem doesn’t lie on a path between original particle positions and such a local optimum, then this convergence behavior prevents effective search for the global optimum. Many of the particles are wasting computational effort in seeking to move in the same direction (towards the local optimum already discovered), whereas better results may be obtained if various particles explore other possible search directions. Al-Kazemi and Mohan[12] have recently developed an algorithm called the Multiphase PSO that explores this idea by allowing some particles to move away from the best position discovered so far. This paper explores another alternative in which the particles are influenced by other particles, not just moving towards or away from the best position discovered so far.

The socio-cognitive learning process defined in the standard PSO is based on a particle’s own experience and the experience of the most successful particle. The FDR-PSO algorithm adds a new dimension to this approach: each particle also learns from the experience of the neighboring particles that have a better fitness than itself. We believe that this modification of PSO is consistent with the actual dynamics of organisms in groups. This approach results in changes in the velocity update equations, although the position update equations remain unchanged.

Attempts to introduce the effects of multiple other (neighboring) particles on each particle must face the possibility of crosstalk effects encountered in neural network learning algorithms. In other words, the pulls experienced in the directions of multiple other particles may mostly cancel each other, reducing the possible benefit of all the associated computations. To counteract this possibility, the FDR-PSO algorithm selects only one other particle at a time when updating each velocity dimension. This particle is chosen to satisfy two criteria:

1. It must be near the particle being updated.
2. It should have visited a position of higher fitness.

We have experimented with several possible ways of selecting particles that satisfy these criteria, without significant difference in the performance of the resulting algorithm. The simplest and most robust variation was to update each velocity component by selecting a particle that maximizes the ratio of the fitness difference to the one-dimensional distance. In other words, the $d$th dimension of the $i$th particle’s velocity is updated using a particle called the $nbest$, with prior best position $P_j$, chosen to maximize

$$
\frac{\text{Fitness}(P_j) - \text{Fitness}(X_i)}{|P_{jd} - X_{id}|}
$$

where $[...]$ denotes the absolute value, and it is presumed that the fitness function is to be maximized. The above expression is called the Fitness-Distance-Ratio, suggesting the name FDR-PSO for the algorithm; for a minimization problem, we would instead use \((\text{Cost}(P_j) - \text{Cost}(X_i))\) in the numerator of the above expression.

This version of the algorithm was found to be more successful than variations such as selecting a single particle in whose direction all velocity components are updated.

The pseudocode for this algorithm is given below. Note that the particle’s velocity update is influenced by three factors:

1. Previous best experience i.e. $pbest$ of the particle.
2. Best global experience i.e. $gbest$, considering the best $pbest$ of all particles.
3. Previous best experience of the “best nearest” neighbor i.e. $nbest$.
The velocity update equation becomes:
\[
V_{id}^{t+1} = \omega \times V_{id}^{t} + \psi_1 \times (p_{id} - X_{id}) + \psi_2 \times (p_{gd} - X_{id}) + \psi_3 \times (p_{nd} - X_{id})
\] (4)

The position update equation remains the same as in (2).

Algorithm FDR-PSO:
For \( t = 1 \) to the max. bound of the number on generations,
For \( i = 1 \) to the population size,
For \( d = 1 \) to the problem dimensionality,
Let \( n_{best} \) be the particle \( j \) that maximizes
\[
\frac{\text{Fitness}(P_j) - \text{Fitness}(X_i)}{p_{id} - X_{id}}
\]
Apply the velocity update equation:
\[
V_{id}^{t+1} = \omega \times V_{id}^{t} + \psi_1 \times (p_{id} - X_{id}) + \psi_2 \times (p_{gd} - X_{id}) + \psi_3 \times (p_{nd} - X_{id})
\]
where \( P_i \) is the best position visited so far by \( X_i \),
and \( P_g \) is the best position visited so far by any particle;
Limit magnitude:
\[
V_{id}^{t+1} = \min(V_{max} \times \max(-V_{max}, V_{id}^{t+1})))
\]
Update Position:
\[
X_{id}^{t+1} = \min(Max_{d} \times \max(-\text{Min}_{d} X_{id}^{t} + V_{id}^{t+1})))
\]
End-for-d;
Compute fitness of \( X_{id}^{t+1} \);
If needed, update historical information regarding \( P_i \)
and \( P_g \);
End-for-i;
Terminate if \( P_g \) meets problem requirements;
End-for-t;
End algorithm.

III. EXPERIMENTAL SETTINGS AND BENCHMARK PROBLEMS.

Problems are encoded using direct real-valued parameter representations. The implementations are scalable, i.e., the dimensions of the functions are adjustable via a single parameter used in the function [11]. The following functions have been used for evaluation of FDR-PSO.

1. De Jong’s function 1
\[
f(x) = \sum_{i=1}^{n} x_i^2
\] (5)
where \(-5.12 \leq x_i \leq 5.12\)
Global minimum: \( x_i = 0, f(x) = 0 \)
This simple test function is continuous, convex and unimodal.

2. Axis parallel hyper-ellipsoid
\[
f(x) = \sum_{i=1}^{n} i \times x_i^2
\] (6)
where \(-5.12 \leq x_i \leq 5.12\)
Global minimum: \( x_i = 1, f(x) = 0 \)
This function, also known as weighted sphere model, is continuous, convex and unimodal.

3. Sum of different powers
\[
f(x) = \sum_{i=1}^{n} \left| x_i \right|^{i+1}
\] (7)
where \(-1 \leq x_i \leq 1\)
Global minimum: \( x_i = 1, f(x) = 0 \)
This is a commonly used unimodal test function.

4. Rotated hyper-ellipsoid
\[
f(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2
\] (8)
where \(-65.536 \leq x_i \leq 65.536\)
Global minimum: \( x_i = 0, f(x) = 0 \)

5. Rosenbrock’s Valley (Banana function).
\[
f(x) = \sum_{i=1}^{n-1} 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2
\] (9)
where \(-2.048 \leq x_i \leq 2.048\)
Global minimum: \( x_i = 1, f(x) = 0 \)
Rosenbrock’s valley is a classic optimization problem. The global optimum is inside a long, narrow, parabolic shaped valley. To find the valley is trivial, however convergence to the global optimum is difficult. Hence this problem is often used in assessing the performance of the optimization algorithms [11].

6. Griewangk’s Function
\[
f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1
\] (10)
where \(-600 \leq x_i \leq 600\)
Global minimum: \( x_i = 0, f(x) = 0 \)
Griewangk’s function is also a multimodal problem and the location of minima are regularly distributed [11].
Table I shows the parameter values used for PSO and the FDR-PSO. The social learning rate and the cognitive learning rate are equal in both PSO and FDR-PSO. The PSO doesn’t have the third parameter, found in FDR-PSO. The selection of different parameters in a PSO is application-specific. The parameters chosen here are the simplest. The inertia parameter is decremented with the number of iterations, as in [15]:

\[
\omega^{(i+1)} = \frac{\omega^{(i)} - 0.4 \times (gsize - i)}{gsize + 0.4}
\]  

(11)

where gsize is the total number of generations for which the algorithm runs and i is the present generation number.

**TABLE I.: PARAMETER VALUES USED IN PSO AND FDR-PSO**

<table>
<thead>
<tr>
<th>Parameters/Algorithm</th>
<th>(\psi_1)</th>
<th>(\psi_2)</th>
<th>(\psi_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>FDR-PSO</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

We also performed experiments with the third parameter \(\psi_3 = 1\), results were almost as good as with \(\psi_3 = 2\).

Table II shows the number of particles and the number of generations for which each optimization algorithm is run. All the simulations were performed for 30 trials and the averages over 30 trials have been plotted in graphs and tables. Each algorithm was terminated after 1000 generations, in each simulation.

**TABLE II.: POPULATION SIZES AND PROBLEM DIMENSIONALITY IN VARIOUS EXPERIMENTS FOR PSO AND FDR-PSO**

<table>
<thead>
<tr>
<th>Function</th>
<th>Population Size</th>
<th>Generations</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>De Jong’s</td>
<td>10</td>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>Axis Parallel hyper-Ellipsoid</td>
<td>10</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>Sum of Powers</td>
<td>10</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>Rotated hyper-Ellipsoid</td>
<td>10</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>Rosenbrock’s</td>
<td>10</td>
<td>1000</td>
<td>2</td>
</tr>
<tr>
<td>Griewangk’s</td>
<td>10</td>
<td>1000</td>
<td>10</td>
</tr>
</tbody>
</table>

**IV. RESULTS**

Figures 1 through 14 present the results on the optimization functions defined in the previous sections. The graphs show results averaged over 30 trials. In each trial, the population is randomly initialized and the same population is used for PSO and FDR-PSO.
Fig. 4. Best fitness vs. Average Fitness using the Standard PSO for Rotated Hyper-Ellipsoid

Fig. 5. Best fitness vs. Average Fitness using the FDR-PSO for Rotated Hyper-Ellipsoid

Fig. 6. Standard PSO vs. FDR-PSO for Griewank’s Function

Fig. 7. Best fitness vs. Average Fitness using the Standard PSO for Griewank’s Function

Fig. 8. Best fitness Vs. Average Fitness using the FDR-PSO for Griewank’s Function

Fig. 9. Standard PSO vs. FDR-PSO for Sum of Powers function
Fig. 10. Best fitness vs. Average Fitness using the Standard PSO for Sum of Powers function

Fig. 11. Best fitness vs. Average Fitness using the FDR-PSO for Sum of Powers function

Fig. 12. Standard PSO vs. FDR-PSO for Rosenbrock’s Valley

Fig. 13. Best fitness vs. Average Fitness using the Standard PSO for Rosenbrock’s Valley

Fig. 14. Best fitness vs. Average Fitness using the FDR-PSO for Rosenbrock’s Valley

<table>
<thead>
<tr>
<th>Optimization Function</th>
<th>Minima Achieved FDR-PSO</th>
<th>PSO</th>
<th>Convergence FDR-PSO</th>
<th>Convergence PSO</th>
<th>Iterations FDR-PSO</th>
<th>Iterations PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>De Jong’s</td>
<td>2.2415 e-006</td>
<td>0.0156</td>
<td>Never Converged</td>
<td></td>
<td>804</td>
<td></td>
</tr>
<tr>
<td>Axis Parallel Hyper-Ellipsoid</td>
<td>1.8740e-018</td>
<td>2.1546e-007</td>
<td>Never Converged</td>
<td></td>
<td>820</td>
<td></td>
</tr>
<tr>
<td>Rotated Ellipsoid</td>
<td>5.7622 e-007</td>
<td>0.0038</td>
<td>Never Converged</td>
<td></td>
<td>813</td>
<td></td>
</tr>
<tr>
<td>Rosenbrock’s</td>
<td>4.0697 e-012</td>
<td>5.3273e-008</td>
<td>Never Converged</td>
<td></td>
<td>869</td>
<td></td>
</tr>
<tr>
<td>Griewangk’s</td>
<td>1.1842 e-016</td>
<td>1.3217e-008</td>
<td>Never Converged</td>
<td></td>
<td>680</td>
<td>713</td>
</tr>
<tr>
<td>Sum of Powers</td>
<td>9.3621 e-034</td>
<td>2.3626e-011</td>
<td>Never Converged</td>
<td></td>
<td>806</td>
<td></td>
</tr>
</tbody>
</table>
“Convergence” in Table III refers to the settling of particles into positions from which they do not budge in further iterations, so that the algorithm stagnates and no better solutions are discovered thereafter. In this sense, convergence implies wastage of computational resources. A special case is when all the particles converge to the same position; in many evolutionary algorithms, convergence is indicated when the average fitness of the population becomes identical to the best fitness, although this is not a strict logical relation.

As shown in Figures 1-14 and Table III, the new FDR-PSO algorithm outperforms the classic PSO algorithm on each of the benchmark problems on which we have conducted experiments so far. In each case, we find that the original PSO algorithm performs well in initial iterations but fails to make further progress in later iterations.

Figures 4, 7, 10, and 13 indicate possible reasons for the same: in PSO the population diversity is rapidly lost, with particles reaching the same fitness (since best fitness becomes almost identical to the average fitness), and probably converged to the same position, after which particles cease to move. This doesn’t happen in FDR-PSO as shown in figures 5, 8, 11 and 14, where the best and average fitness continue to differ for many more iterations than in the old PSO algorithm. This effect was most marked for Rosenbrock’s valley function, as shown in figure 14. Thus, FDR-PSO is much less likely than PSO to get stuck in local optima of the function being optimized.

V. DISCUSSION

This paper has proposed a new variation of the particle swarm optimization algorithm called FDR-PSO. The new algorithm introduces a new term into the velocity component update equation: particles are moved in the direction of a nearby particle’s best prior position, preferring positions of higher fitness. The implementation of this idea is simple, based on computing and maximizing the relative fitness-distance-ratio. The new algorithm is shown to outperform PSO on many benchmark problems, being less susceptible to premature convergence, and less likely to be stuck in local optima.

In future work, we plan to investigate more fully the effects of varying parameter values such as $\psi_1, \psi_2, \psi_3$ on performance of the FDR-PSO algorithm. Preliminary results show that only slight variations are then obtained in the performance of the algorithm.

Several interesting variations of the PSO algorithm have recently been proposed by researchers. Many of these PSO improvements (described below) are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm, and probably can be added on to the FDR-PSO, a possibility we intend to explore in future work.

Although we have not implemented all these algorithms and conducted exhaustive comparative evaluations, the following paragraphs report preliminary comparisons using the results reported in the publications cited below; these comparisons suggest that FDR-PSO outperforms many of the recent improvements of the PSO algorithm.

1. Van Den Bergh and Engelbrecht [16] [17] have suggested a cooperative version of PSO, called the “Split PSO” in which each component of the solution vector is separately optimized using a 1-dimensional PSO; however separately tuning individual problem parameters in this manner is not always successful, particularly when these parameters interact in a non-additive manner. A related concept of “epistasis” has been investigated extensively in the Genetic Algorithms literature.

2. Lovebjerg and Krink [18] have explored extending the PSO with “Self-Organized Criticality,” aimed at improving population diversity. In their algorithm, a measure, describing how close to each other are particles in the swarm, is used to determine whether to relocate particles. This is akin to the “random restart” approach to increasing population diversity in genetic algorithms when premature convergence is detected or suspected. Lovebjerg, Rasmussen, and Krink also proposed an idea of splitting the population of particles into subpopulations and hybridizing the algorithm borrowing the concepts from Genetic Algorithms. The results reported for Griewangk’s function with a dimensionality of 30, swarm size of 10 and 1500 iterations, FDR-PSO gave a minima of 3.7x10^{-5} whereas the Split-PSO reached a minima of 2.26x10^{-2}.

3. The random restart mechanism has also been proposed under the name of “PSO with Mass Extinction” [19] by Xie, Zhang, and Yang. The latter researchers have also explored increasing diversity by increasing the randomness associated with the particle velocity and position updates [20], thereby discouraging swarm convergence, in the “Dissipative PSO” algorithm. In [19], the results reported for Griewangk's function for a dimensionality of 10, swarm size 20 and 1000 iterations are:

HPSO$_1$ - 0.09100
HPSO$_2$ - 0.08626
Our experiments for the same problem result in a minima of $1.18 \times 10^{-16}$.

4. Riget and Vesterstrom[21] have explored yet another variation called the “Attraction-Repulsion-PSO (ARPSO)”, also aimed at increasing population diversity. This algorithm cycle the swarm through multiple phases in which particles are alternatively attracted and repelled by the best positions found so far.

5. A similar idea was implemented in Al-kazemi and Mohan’s “Multiphase PSO (MuPSO)” [12], in which different groups of particles step through different phases, with each group pursuing a different direction (towards or away from the best solution found so far); MuPSO also hybridizes PSO with a hill-climbing component.

VI. REFERENCES


