2007

An estimation of distribution improved particle swarm optimization algorithm

Raghavendra V. Kulkarni

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Follow this and additional works at: http://scholarsmine.mst.edu/faculty_work

Part of the Electrical and Computer Engineering Commons

Recommended Citation
http://scholarsmine.mst.edu/faculty_work/974

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. For more information, please contact weaverjr@mst.edu.
An Estimation of Distribution Improved Particle Swarm Optimization Algorithm

R. V. Kulkarni and G. K. Venayagamoorthy

Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering
University of Missouri-Rolla, MO 65409, USA
1arvie@ieee.org & 2gkumar@ieee.org

Abstract

PSO is a powerful evolutionary algorithm used for finding global solution to a multidimensional problem. Particles in PSO tend to re-explore already visited bad solution regions of search space because they do not learn as a whole. This is avoided by restricting particles into promising regions through probabilistic modeling of the archive of best solutions. This paper presents hybrids of estimation of distribution algorithm and two PSO variants. These algorithms are tested on benchmark functions having high dimensionalities. Results indicate that the methods strengthen the global optimization abilities of PSO and therefore, serve as attractive choices to determine solutions to optimization problems in areas including sensor networks.

1. INTRODUCTION

Particle swarm optimization algorithm (PSO), models the dynamics of societies of biological specimen like birds, insects and fish. It is a population based optimization technique in which a collection of test solutions interact with each other and search for the best solution to the given problem [1].

PSO consists of a population (or swarm) of particles, each of which represents an n dimensional potential solution. Particles are assigned random initial positions and they change their positions iteratively to reach the global optimal solution. The direction of position change is influenced by both particle’s own experience and the knowledge the particle acquires from the flock. Each particle is evaluated using a fitness function, which indicates how close the particle is to the optimal solution. It is calculated fitness value which determines solutions to optimization problems in areas including sensor networks.

FOR each particle i
   FOR each dimension d
      Initialize position xid(k) randomly within permissible range
      Initialize velocity vid(k) randomly within permissible range
   END FOR
   END FOR
   Iteration k=1
   DO
      FOR each particle i
         Calculate fitness value
         IF the fitness value is better than p_bestid in history
            Set current fitness value as the p_bestid
         END IF
      END FOR
      FOR each dimension d
         Calculate velocity according to the equation
         \[ v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot \text{rand} \cdot (p_{id} - x_{id}) + c_2 \cdot \text{rand} \cdot (g_{id} - x_{id}) \] (1)
         Update particle position according to the equation
         \[ x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \] (2)
      END FOR
   END FOR
   k=k+1
   WHILE maximum iterations or minimum error criteria are not attained

Fig. 1. Pseudocode for PSO

PSO has found extensive applications in many optimization problems. In sensor networks, it has been used for cluster formation [2], optimal multicast routing [3], and distributed sensor placement problems [4]. Maximum likelihood estimation of target position [5] and sink node path optimization [6] are the other problems that have been addressed with PSO. A PSO variant has been applied in wavelength detection in FGB sensor network [7]. PSO has been used for odor source localization in mobile sensor networks [8].

In spite of its advantages like low computational complexity, the PSO suffers from the problem of premature convergence. This is overcome with a mutation operator with adaptive probability, and by replacing particles flying out of the solution space by newly generated random particles during the search process. The variant of PSO that uses adaptive mutation and regeneration is called Improved PSO (IPSO) [9].
In the classical PSO, particles depend on their individual memory and peer influence to explore the search space. However, the swarm as a whole does not use its collective experience (represented by the array of previous best positions) to guide its search. This causes re-exploration of already known bad regions in the search space. This paper proposes an approach in which swarm's collective memory is used to guide the particle’s movement towards the estimated good regions in the search space.

This paper presents two hybrid versions of PSO that allow a particle swarm to estimate the distribution of promising solution regions and thus learn through the information assimilated during the process of optimization. This distribution is used to keep the particles within the promising solution regions. This algorithm is fused with two versions of PSO, namely classical PSO and IPSO. The estimation of the distribution is done by means of a mixture of normal distributions of previous best solutions. These hybrids borrow ideas from recent developments in Ant Colony Optimization (ACO) in which an archive of solutions is used to select the next point to explore in the search space.

PSO and the two hybrid versions of PSO proposed here are tested on five benchmark test functions. The rest of the paper is organized as follows: Section 2 describes the background of the estimation of distribution PSO algorithm (EDPSO). Section 3 covers the details of the estimation of distribution improved PSO algorithm (EDIPSO). Numeric simulation and results are presented in section 4 and conclusions are given in section 5.

2. ESTIMATION OF DISTRIBUTION PSO (EDPSO) ALGORITHM

Estimation of distribution algorithms (EDA) use information obtained during optimization to build probabilistic models of distribution of good solution regions and use this information to produce new solutions. EDAs yield fast convergence to global optimal solution because they approximate the joint probability distribution that characterizes the problem. A comprehensive comparison of some best-known EDA algorithms is given in [10]. This paper uses two hybrids, which progress like PSO algorithms but model the joint probability distribution in order to constrain particles in better areas of search space.

Ant Colony Optimization (ACO) is another popular swarm intelligence algorithm. This algorithm is used for combinatorial optimization problems. A recent development of ACO that is aimed at continuous optimization is called ACO<sub>R</sub> [11]. This algorithm approximates the joint probability distribution, one dimension at a time, by using a mixture of weighted Gaussian functions. The weights represent quality of different search regions and thus learn through the information assimilated during the process of optimization. This has the same value for all dimensions because it is based on relative quality of complete solutions. In each iteration, solutions are ranked and weights are determined using (3),

\[ w_j = \frac{1}{qm \sqrt{2\pi}} e^{-\frac{(l-i)^2}{2(qm)^2}} \]  

where \( q \) is the parameter that determines the degree of preference of good solutions. With a small value of \( q \), best solutions are strongly preferred over weaker solutions to guide the search [12].

![Joint Probability Distribution based on weighted Gaussians](image)

Fig. 2. Joint probability distribution based on weighted Gaussians

Because the algorithm samples a mixture of Gaussians, one will need to select one Gaussian function from the kernel probabilistically. The probability of choosing the \( l \)th Gaussian function is computed using (4),

\[ P_l = \frac{w_l}{\sum_{j=1}^{m} w_j} \]  

The standard deviation of the Gaussian functions is computed using (5),

\[ \sigma_l = \frac{\bar{z} \sum_{j=1}^{m} |s_{ij} - s_{lj}|}{m-1} \]  

where \( \bar{z} \) is a parameter that allows algorithm to balance its exploration-exploitation behavior. This has the same value for all dimensions. ACO<sub>R</sub> samples a Gaussian function and generates a new solution component in every iteration. This paper borrows the idea from ACO<sub>R</sub> [11].

In estimation of distribution PSO (EDPSO) hybrid, ACO<sub>R</sub> is fused with PSO in order to exploit the useful properties of both the algorithms. The \( p_{best} \) matrix is used here as the archive of solution over which ACO<sub>R</sub> builds its probabilistic model. The EDPSO algorithm progresses as the normal PSO does. For
every particle in the swarm, two particles are generated, one using PSO and another using estimation of distribution. In each iteration, the location to which a particle will be moved is determined using PSO position update equation. Such a particle is names as PSO version of the particle. In addition, a Gaussian distribution function is probabilistically chosen from the kernel and a new particle is produced by sampling it in all $n$ dimensions. This gives the EDA version of the particle. The fitness functions are evaluated for both versions of a particle. The particle that exhibits the better objective function is selected and a new particle is produced by sampling it in all dimensions. This gives the EDA version of the particle. The particle regeneration.

3. ESTIMATION OF DISTRIBUTION IMPROVED PSO (EDIPSO) ALGORITHM

As seen in equation (1), the velocity update of the particle consists of three parts: the first term is inertia of particles; the second term is cognitive acceleration which represents the particle's own experiences; and the third term is social acceleration which represents the social interaction between the particles. From this, it can be reasoned that when a particle’s current position coincides with the global best position, the particle will leave this place only if the inertia weight $w$ and its current velocity $v$ are not equal to 0. If the particles’ current velocities are very close to 0, then the particles will not move if they get caught up with the best particle. This means that all the particles will converge to the best position $g_{best_d}$ if this position is not the global best, then this phenomenon leads to premature convergence.

Improved PSO (IPSO) uses adaptive mutation to avoid premature convergence [9]. If $x=x_1,x_2,...x_n$ is the particle chosen with mutation probability $P_m$, then the mutation result of this particle is

$$x'_d = g_{best_d} + 0.50 \times \text{randn}(g_{best_d})$$

In the mutation operation, the mutation probability $P_m$ is dynamically adjusted according to the diversity in the swarm. The ratio between mean and the maximum of the fitness function of all particles in an iteration is used to measure the diversity $div$, such that $0 < div < 1$. If $div \approx 1$, it means that all the particles have gathered at the same position. Under these circumstances, the mutation probability should be increased to allow more particles to search in different unexplored zones. On the contrary, if $div<<1$ it indicates that there is a great diversity of particles in the swarm, in which case the $P_m$ must be reduced to avoid a basically random search. The pseudocode for the dynamic adaptation of the mutation is shown in Fig. 4.

![Fig. 4. Pseudocode for dynamic adaptive mutation](image)

The standard PSO algorithm generally uses boundary condition to constrain particles in the search region. On the other hand, IPSO algorithm produces the same number of random particles to replace the particles that fly out of the search space [2] [9]. The pseudocode for the regeneration is given in Fig 5. Pseudocode for EDIPSO algorithm is shown in Fig. 6.

![Fig. 5. Pseudocode for regeneration](image)

The EDIPSO proposed here is a hybrid of IPSO and the ACO. The difference between EDIPSO and EDIPSO lies in the fact that the latter uses both dynamic adaptive mutation and particle regeneration.
FOR each particle $i$
    FOR each dimension $d$
        Initialize position $x_{id}$ randomly within permissible range
        Initialize velocity $v_{id}$ randomly within permissible range
    END FOR
END FOR

Iteration $k=1$
DO
    FOR each particle $i$
        Calculate fitness value
        IF the fitness value is better than $p_{bestid}$ in history
            Set current fitness value as the $p_{bestid}$
        END IF
    END FOR

Choose the particle having the best fitness value as the $g_{best}$
DO
    FOR each dimension $d$
        FOR each particle $i$
            Calculate velocity according to the equation
            \[ v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot rand(x_{pbestid} - x_{id}) + c_2 \cdot rand(x_{gbestd} - x_{id}) \]
            Update particle position according to
            \[ x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \]
            Apply adaptive mutation with probability $p_m$
            Compute weights if Gaussian functions in kernel $w_i$
            Select a Gaussian function $g_i$ from kernel according to $p_i$
            Apply regeneration if a particle flies out of search space
        END FOR
    END FOR
END FOR

END FOR

WHILE maximum iterations or minimum error criteria are not attained

Fig 6. Pseudocode for EDIPSO algorithm

4. NUMERICAL SIMULATION AND RESULTS

All simulations are carried out on the same computer using Matlab. Relative performance of PSO, EDPSO and EDIPSO is tested to minimize five standard benchmark functions given in Table I. All the functions have global minima at zero.

Initial assignment of weights and maximum values of velocities and positions are taken as shown in Table II. In all of the functions, dimensionalities $n=50$ and $n=100$ are tested. The parameters chosen are:

- Number of particles in the swarm: 30
- Inertia weight is computed by (12) where $w_{max} = 0.9$ and $w_{min} = 0.4$

\[ w_{max} = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \]

- Acceleration constants $c_1 = 2.0$ and $c_2 = 2.0$
- Initial mutation probability in EDIPSO, $P_m = 0.08$
- $P_{max} = 0.15$, $P_{min} = 0.01$
- For $n=50$, Maximum iterations $iter_{max} = 5000$
- For $n=100$, Maximum iterations $iter_{max} = 10000$
- In EDPSO and EDIPSO, $q=0.05$ and $\zeta=0.85$
- PSO, EDPSO and EDIPSO algorithms are tested with the same set of random initial particles.

Each algorithm is tested for 20 trials. Average fitness and standard deviation of fitness are computed. The results obtained are presented in Table III.

Table I. List of benchmark functions on which PSO, EDPSO and EDIPSO are tested

<table>
<thead>
<tr>
<th>Function</th>
<th>Initialization range</th>
<th>$V_{max}$</th>
<th>$X_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>$(50,100)^n$</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Rosenbrock Function</td>
<td>$(15,30)^n$</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Rastrigrin</td>
<td>$(2.56, 5.12)^n$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Griewank</td>
<td>$(-50,50)^n$</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Ackley</td>
<td>$(-32,32)^n$</td>
<td>10</td>
<td>32</td>
</tr>
</tbody>
</table>

Fig. 7. Results of Optimization of Sphere Function
Results of optimization of benchmark functions reveal that in all functions except Rastrigrin, EDPSO algorithm produces better quality of solutions than classical PSO, both in terms of fitness and standard deviation. This is because particles are guided towards the better solution zones due to probabilistic modeling. Further, it is observed that the concepts of adaptive mutation and regeneration yield more efficient search for global optimal solution in EDIPSO algorithm. Best solutions determined by EDIPSO are several orders lesser than the best solutions determined by EDPSO algorithm.

EDIPSO algorithm exhibits remarkable speed in optimizing all the functions tested. Besides, it produces consistent performance over trial runs as indicated by the small standard deviation. This makes EDIPSO algorithm one of the very promising algorithms available for optimization of multimodal functions.

### 5. CONCLUSIONS AND FUTURE WORK

In this paper, two versions of EDA–PSO hybrid are introduced. Results of optimization of benchmark functions indicate that EDPSO and EDIPSO have abilities to find better quality of solutions than that of PSO. This renders these algorithms attractive for optimization problems in sensor networks like cluster formation, multicast routing, distributed sensor placement and sink node path optimization.
Different EDIPSO optimization parameters are required for solving different problems in practical application, such as the number of agents (individuals), weight factors and, acceleration factors and the limits for change in velocity. Sensitivity analysis of optimization parameters for finding the best solutions is one of the future works. Further scope for research lies in hybrids of other forms of EDA and PSO and their applications to specific optimization problems in sensor networks.

REFERENCES


