Information Sharing Strategy among Particles in Particle Swarm Optimization Using Laplacian Operator

Jagdish Chand Bansal, Kusum Deep, Kalyan Veeramachaneni, Lisa Osadciw

Abstract— Particle Swarm Optimization (PSO) has been extensively used in recent years for the optimization of nonlinear optimization problems. Two of the most popular variants of PSO are PSO-W (PSO with inertia weight) and PSO-C (PSO with constriction factor). Typically particles in swarm use information from global best performing particle, gbest and their own personal best, pbest. Recently, studies have focused on incorporating influences of other particles other than gbest. In this paper, we develop a methodology to share information between two particles using a Laplacian operator designed from Laplace probability density function. The properties of this operator are analyzed. Two particles share their positional information in the search space and a new particle is formed. The particle, called as Laplacian particle, replaces the worst performing particle in the swarm. Using this new operator, this paper introduces two algorithms namely Laplace Crossover PSO with inertia weight (LXPSO-W) and Laplace Crossover PSO with constriction factor (LXPSO-C). The performance of the newly designed algorithms is evaluated with respect to PSO-W and PSO-C using 15 benchmark test problems. The empirical results show that the new approach improves performance measured in terms of efficiency, reliability and robustness.

I. INTRODUCTION

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart in 1995 [1], solves problems in many fields, especially engineering and computer science. The power of the technique is its fairly simple computations and sharing of information within the algorithm as it derives its internal communications from the social behavior of individuals. The individuals, called particles henceforth, are flown through the multi-dimensional search space with each particle representing a possible solution to the multi-dimensional optimization problem. Each solution’s fitness is based on a performance function related to the optimization problem being solved.

The movement of the particles is influenced by two factors using information from iteration-to-iteration as well as particle-to-particle. As a result of iteration-to-iteration information, the particle stores in its memory the best solution visited so far, called pbest, and experiences an attraction towards this solution as it traverses through the solution search space. As a result of the particle-to-particle information, the particle stores in its memory the best solution visited by any particle, and experiences an attraction towards this solution, called gbest, as well. The first and second factors are called cognitive and social components, respectively. After each iteration, the pbest and gbest are updated for each particle if a better or more dominating solution (in terms of fitness) is found. This process continues, iteratively, until either the desired result is converged upon, or it’s determined that an acceptable solution cannot be found within computational limits.

For an n-dimensional search space, the i-th particle of the swarm is represented by a n-dimensional vector, \( X_i = (x_{i1}, x_{i2}, ..., x_{in})^T \). The velocity of this particle is represented by another n-dimensional vector \( V_i = (v_{i1}, v_{i2}, ..., v_{in})^T \). The previously best visited position of the i-th particle is denoted as \( P_i = (p_{i1}, p_{i2}, ..., p_{in})^T \). 'g' is the index of the best particle in the swarm. The velocity of the i-th particle is updated using the velocity update equation given by

\[
V_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}),
\]

and the position is updated using

\[
x_{id} = x_{id} + v_{id}
\]

where \( d = 1, 2, ..., n \) represents the dimension and \( i = 1, 2, ..., S \) represents the particle index. \( S \) is the size of the swarm and \( c_1 \) and \( c_2 \) are constants, called cognitive and social scaling parameters respectively (usually, \( c_1 = c_2 \); \( r_1 \) and \( r_2 \) are random numbers drawn from a uniform distribution). Equations (1) and (2) define the classical version of PSO algorithm. A constant, \( V_{max} \), was introduced to arbitrarily limit the velocities of the particles and improve the resolution of the search. The maximum velocity \( V_{max} \), serves as a constraint to control the global exploration ability of particle swarm. Further, the concept of an inertia weight was developed to better control exploration and exploitation. The motivation was to be able to eliminate the need for \( V_{max} \). The inclusion of an inertia weight in the particle swarm optimization algorithm was first reported in the literature in 1998 [2]. The resulting velocity update equation becomes:

\[
v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})
\]

Eberhart and Shi, [3] indicate that the optimal strategy is to initially set \( \omega \) to 0.9 and reduce it linearly to 0.4, allowing
initial exploration followed by acceleration toward an improved global optimum.

Clerc has introduced a constriction factor, $\chi$ [4], which improves PSO’s ability to constrain and control velocities. $\chi$ is computed as:

$$\chi = \frac{2}{2 - \phi - \sqrt{\phi(\phi - 4)}}$$ (4)

where $\phi = c_1 + c_2$, $\phi > 4$, and the velocity update equation is then

$$v_{id} = \chi \times \left( v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \right)$$ (5)

Eberhart and Shi, [3] found that $\chi$, combined with constraints on $V_{max}$, significantly improved the PSO performance.

In classical particle swarms, the particle moves using the information from its previous best and the global best particle. Researchers have modified the interaction by incorporating nearest best neighbor using Fitness-Distance-Ratio metric [5]. Recently, a Fitness-Euclidean distance ratio metric [6] has been used to decide the neighbor that will influence the particle in its movement.

Previous research has focused on designing the influences for a particular particle under consideration. In this paper we introduce a crossover operator that develops an information sharing model between any two randomly chosen particles. In [7], a crossover step is added to the standard PSO. The crossover is performed between each particle’s individual best positions. After the crossover, the fitness of the individual best position is compared with that of the two off-springs, and the best one is taken as the new individual best position. The crossover can help the particles jump out of the local optimization by sharing the others’ information. In [8], to improve the solution diversity in PSO, a crossover PSO is introduced. Two particles are selected as parents through tournament selection. After selecting a crossover point randomly, new particles are generated using crossover probability. Both the above techniques perform a crossover by swapping the particles around the crossover point. The positions themselves are not altered. In this paper, we perform crossover and alter the position during this crossover.

The rest of the paper is organized as follows. An operator using Laplacian distribution is presented for PSO in section 2. Analysis of probabilistic nature of this crossover is analyzed. The new particle swarm algorithm incorporating this new operator is presented in section 3. The new operator is applied to two interacting particles resulting in the replacement of the worst particle in the population. In Section 4, the benchmarks and experimental settings are described. Results are presented in Section 5 followed by conclusions and future work in Section 6.

II. PROPERTIES AND ANALYSIS OF LAPLACIAN CROSS OVER

In this section, a crossover for PSO is proposed which uses Laplace Distribution. This crossover operator was first introduced for genetic algorithms in [9]. This parent centric operator is called Laplace Crossover (LX) Operator. LX has similar properties like Simulated Binary Crossover Operator (SBX) [10]. The probability density function for Laplace distribution is given by

$$f(x|a, b) = \frac{1}{2b} \exp\left(-\frac{|x-a|}{b}\right), -\infty < x < \infty,$$ (6)

and the cumulative density function is given by

$$F(x|a, b) = \begin{cases} 
\frac{1}{2} \exp\left(-\frac{|x-a|}{b}\right), & x \leq a \\
1 - \frac{1}{2} \exp\left(-\frac{|x-a|}{b}\right), & x > a 
\end{cases}$$ (7)

where $a \in R$ is the location parameter and $b > 0$ is the scale parameter. In Figure 1, the Laplacian probability density function is plotted for different scale parameter. We fix the location parameter $a=0$ and plot for two different values of $b$, i.e., $b=0.5$ and $b=1$. We see that the scale parameter determines the width of the distribution. In Figure 2, we fix the scale parameter and change the plot the probability density function for two location parameters, $a=0$ and $a=2$. 

Fig.1. Impact of scale change on Laplace Density function

Fig.2. Impact of location change on Laplace Density Function
Two off-springs \( y_1 = (y_{11}, y_{12}, \ldots, y_{1n}) \) and \( y_2 = (y_{21}, y_{22}, \ldots, y_{2n}) \) are generated from a pair of parents \( x_1 = (x_{11}, x_{12}, \ldots, x_{1n}) \) and \( x_2 = (x_{21}, x_{22}, \ldots, x_{2n}) \) in the following manner.

1. First, a uniformly distributed random number \( u_i \in (0,1) \) is generated. Then, from Laplace distribution function, the ordinate \( \beta_i \) is calculated so that the area under the probability curve excluding area from \( a \) (location parameter) to \( \beta_i \) is equal to chosen random number \( u_i \). The calculation is carried out in the following way:

First consider \( \beta_i \) to be on the right side of \( a \), then

\[
\beta_i = 1 - \int_a^\beta \frac{1}{2b} \exp \left( -\frac{|x-a|}{b} \right) \, dx.
\]

Since \( a \leq x \leq \beta_i \), so \( |x-a| = (x-a) \)

\[
u_i = 1 - \int_a^{\beta_i} \frac{1}{2b} \exp \left( \frac{x-a}{b} \right) \, dx = 1 - \frac{1}{2} \left[ \exp \left( \frac{\beta_i - a}{b} \right) - 1 \right] = 2u_i - 1 = \exp \left( \frac{-\beta_i - a}{b} \right) \Rightarrow \beta_i = a - b \log_e (2u_i - 1)
\]

Similarly, when \( \beta_i \) is considered to be on the left side to \( a \), then

\[
u_i = 1 - \int_a^{\beta_i} \frac{1}{2b} \exp \left( \frac{a-x}{b} \right) \, dx = 1 - \frac{1}{2} \left[ \exp \left( \frac{\beta_i - a}{b} \right) - 1 \right] = 2u_i - 1 = \exp \left( \frac{\beta_i - a}{b} \right) \Rightarrow \beta_i = a - b \log_e (2u_i - 1)
\]

Thus,

\[
\beta_i = \begin{cases} 
  a - b \log_e (1 - 2u_i), & u_i \leq \frac{1}{2} \\
  a - b \log_e (2u_i - 1), & u_i > \frac{1}{2} 
\end{cases} \tag{8}
\]

2. The off-springs are then given by the equations

\[
y_{1i} = x_{1i} + \beta_i \frac{x_{1i} - x_{2i}}{n} \tag{9}
\]

\[
y_{2i} = x_{2i} + \beta_i \frac{x_{1i} - x_{2i}}{n} \tag{10}
\]

For a fixed value of \( a \) and \( b \), LX dispenses off-springs proportional to the spread of parents i.e. if the parents are near to each other, the off-springs are expected to be near to each other and if the parents are far from each other then the off-springs are likely to be far from each other. This is evident if we subtract (10) from (9) which gives us

\[
\frac{y_{1i} - y_{2i}}{2i} = \frac{x_{1i} - x_{2i}}{2i}.
\]

In this way the proposed crossover operator exhibits self-adaptive behavior. Note that the spiky nature of the Laplacian distribution controls the spread of the off-springs.

### III. LXPSO: PARTICLE SWARM USING LAPLACIAN CROSSOVER

Based on the Laplacian operator described in the previous section, two new particles are formed. The best particle (in terms of fitness) is selected. This new particle, called Laplacian particle, can replace one of the particles from which it is formed or replace the worst performing particle in the swarm. In this paper we analyze swarms behavior if the worst particle (in terms of fitness) is replaced by this Laplacian particle. We call the particle swarm with Laplace crossover as Laplace Crossover PSO (LXPSO). Figure 3 gives the pseudo code for the algorithm.

**Algorithm LXPSO:**

For \( t = 1 \) to the max. bound of the number on iterations, 
For \( i = 1 \) to the swarm size 
For \( d = 1 \) to the problem dimensionality
Apply the velocity update equation; 
Update Position;
End-for-d;

Compute fitness of updated position;
If needed, update historical information for Pi and Pg;

End-for-i;
Select two random particles from the current swarm for interaction. Generate the Laplacian particle as a result of this interaction. Replace the worst particle in the swarm with the Laplacian particle;
Terminate if Pg meets problem requirements;
End-for-t;
End algorithm.

![Fig.3. Pseudo code for LXPSO](image)

We consider two versions of PSO, namely PSO-W (PSO with time varying inertia weight) and PSO-C (PSO with constriction factor) for comparison with LXPSO. Thus two equivalent versions of LXPSO, LXPSO-W (LXPSO with time varying inertia weight) and LXPSO-C (LXPSO with constriction factor) are designed.

### IV. EXPERIMENTAL SETUP AND BENCHMARKS

We test the particle swarm employing the Laplacian crossover operator on the 15 standard benchmark functions. The description of the benchmark problems is given in the following subsection.

#### A. Benchmark Problems

From the standard set of benchmark problems available in the literature, 15 important functions are considered to test
the efficacy of the proposed method. All the problems are composed of continuous variables and have different degree of complexity and multimodality. The set of test functions include unimodal and multimodal functions which are scalable (the problem size can be varied as per the user’s choice). In our experiments, problem size for all problems is set to 30. These are minimization problems having a minimum at 0. The problems are listed in Table 1.

Table I: Benchmark functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Function</th>
<th>Search Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Sphere (De Jong’s f1)</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2$</td>
<td>$-5.12 \leq x_i \leq 5.12$</td>
</tr>
<tr>
<td>2.</td>
<td>De Jong’s f4</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2$</td>
<td>$-5.12 \leq x_i \leq 5.12$</td>
</tr>
<tr>
<td>3.</td>
<td>Griewank</td>
<td>$\min f(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) \right]$</td>
<td>$-600 \leq x_i \leq 600$</td>
</tr>
<tr>
<td>4.</td>
<td>Rosenbrock</td>
<td>Min $f(x) = \sum_{i=1}^{n-1} \left[ 100 (x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2 \right]$</td>
<td>$-30 \leq x_i \leq 30$</td>
</tr>
<tr>
<td>5.</td>
<td>Rastrigin</td>
<td>Min $f(x) = 10a + \frac{1}{n} \sum_{i=1}^{n} \left[ x_i^2 - 10 \cos (2\pi x_i) \right]$</td>
<td>$-5.12 \leq x_i \leq 5.12$</td>
</tr>
<tr>
<td>6.</td>
<td>Ackley</td>
<td>Min $f(x) = -20 \exp \left(-0.02 \sum_{i=1}^{n} x_i^2 \right) - \exp \left(n^{-1} \sum_{i=1}^{n} \cos (2\pi x_i) \right) + 20 + e$</td>
<td>$-30 \leq x_i \leq 30$</td>
</tr>
<tr>
<td>7.</td>
<td>Levy and Montalvo 1</td>
<td>$\min f(x) = \frac{1}{n} \left[ 10 \sin^2 (\pi x_i) + \sum_{i=1}^{n} [(y_i - 1)^2 + 10 \sin^2 (\pi y_i) + (y_i - 1)^2] \right]$</td>
<td>$-10 \leq x_i \leq 10$</td>
</tr>
<tr>
<td>8.</td>
<td>Levy Montalvo 2</td>
<td>Min $f(x) = 0.1 \left[ \sin^2 (3\pi x_i) + \sum_{i=1}^{n} [(y_i - 1)^2 + \sin^2 (3\pi y_i) + (y_i - 1)^2] \right]$</td>
<td>$-5 \leq x_i \leq 5$</td>
</tr>
<tr>
<td>9.</td>
<td>Ellipsoidal 1</td>
<td>Min $f(x) = \frac{1}{n} \left[ \sum_{i=1}^{n} (x_i - 1)^2 \right]$</td>
<td>$-n \leq x_i \leq n$</td>
</tr>
<tr>
<td>10.</td>
<td>Cosine Mixture</td>
<td>Min $f(x) = 0.1n + \frac{1}{n} \sum_{i=1}^{n} x_i^2 - 0.1 \frac{1}{n} \sum_{i=1}^{n} \cos (5\pi x_i)$</td>
<td>$-1 \leq x_i \leq 1$</td>
</tr>
<tr>
<td>11.</td>
<td>Exponential</td>
<td>Min $f(x) = 1 - \exp \left(-0.5 \sum_{i=1}^{n} x_i^2 \right)$</td>
<td>$-1 \leq x_i \leq 1$</td>
</tr>
<tr>
<td>12.</td>
<td>Zakharov’s</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2 + \left[ \frac{1}{n} \sum_{i=1}^{n} \left( \frac{x_i}{2} \right) \right]^2 + \left[ \frac{1}{n} \sum_{i=1}^{n} \left( \frac{x_i}{2} \right) \right]^4$</td>
<td>$-5.12 \leq x_i \leq 5.12$</td>
</tr>
<tr>
<td>13.</td>
<td>Cigar</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{x_i^4}{2} \right] + \left[ \sum_{i=1}^{n} \frac{x_i}{2} \right]^2$</td>
<td>$-10 \leq x_i \leq 10$</td>
</tr>
<tr>
<td>14.</td>
<td>Brown</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{x_i^4}{2} \right] + \left[ \sum_{i=1}^{n} \frac{x_i}{2} \right]^2$</td>
<td>$-1 \leq x_i \leq 4$</td>
</tr>
<tr>
<td>15.</td>
<td>Schewefel 3</td>
<td>Min $f(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{x_i^4}{2} \right] + \left[ \sum_{i=1}^{n} \frac{x_i}{2} \right]^2$</td>
<td>$-10 \leq x_i \leq 10$</td>
</tr>
</tbody>
</table>

B. Particle Swarm Settings

In the literature, different values of parameters are used. In this paper we used the parameter setting as suggested in [11], [12] and [13]. We set swarm size $S = 50$. The inertia weight $w$ is set to reduce linearly from 0.8 to 0.4. Constriction coefficient $c$ is calculated from equation (3). For PSO-C the cognitive and social scaling parameters $c_1$ and $c_2$ are set to 2.8 and 1.3 respectively. And for PSO-W both are set to 2. Global variants of PSO are considered in our experiments. Maximum velocity, $V_{max}$ is set equal to $0.5*(X_{max} - X_{min})$, where $X_{max}$ and $X_{min}$ are the upper and lower bounds of the decision variable. The location and scale parameters $a$ and $b$ for Laplace crossover are 1 and 0.9 respectively (through extensive experiments these parameters are fine tuned). Two criteria are applied to terminate the simulation of the algorithms. The first is if maximum number of iterations, set as 1000, is reached. The second criterion if a minimum error of 0.001 is achieved. For fair comparisons we set the parameters of LXPSO-W and LXPSO-C similar to PSO-C and PSO-W respectively.
V. RESULTS AND DISCUSSION

This section focuses on the efficiency of LXPSO as tested on 15 benchmark functions with 30 dimensions, which are given in Table I. To avoid attributing the optimization results to the choice of a particular initial population and to conduct fair comparisons, we perform each test 100 times, starting from various randomly selected points in the multidimensional search space. The four PSOs (PSO-W, PSO-C, LXPSO-W and LXPSO-C) are implemented in C++. From the recorded simulated results statistical analyses are carried out and presented in Tables II and III. For each algorithm the mean objective function value (Mean OBJ), minimum objective function value (Min OBJ), standard deviation (SD), success rate (SR) and the average number of function evaluations (Average Eval) required to converge to the required function value (Global Optimal + 0.001) were calculated and compared. A success was counted when the condition,

$$f_{\text{min}} - f_{\text{opt}} \leq 0.001$$

was met, where $f_{\text{min}}$ is the best solution found when an algorithm terminates and $f_{\text{opt}}$ is the known global minimum for the problem.

The first goal of the analysis is to analyze the performance benefits of Laplace crossover over the original PSOs for both versions. From Table II, it is clear that from the point of view of success rate LXPSO-W is better than PSO-W in 9 problems, and worse in 1 problem while both perform same in 5 problems. If we consider mean number of function evaluations then LXPSO-W is better than PSO-W in 8 problems, worse in 3 problems and has same performance as PSO-W in 4 problems. From Table III, it is evident that LXPSO-C has higher success rate than PSO-C in 5 problems while same in the remaining 10 problems. From the point of view of average number of function evaluations LXPSO-C is better than PSO-C in 8 problems and worse in 4 problems and have the same performance in 3 problems. Laplace crossover improves the performance of PSO. Now if we compare LXPSO-W and LXPSO-C then we see that LXPSO-C is better than LXPSO-W in 4 problems and same in 11 problems from the point of view of success rate and LXPSO-C requires less number of function evaluations than LXPSO-W in 12 problems and same in 3 problems.

Thus, LXPSO-C performs overall best among all considered methods.

In order to compare the relative performance of PSO-W, PSO-C, LXPSO-W and the LXPSO-C algorithm, the value of a performance index ($PI$) [9], [14], is computed for these four algorithms. This index gives a weighted importance to the success rate; the average number of function evaluations for the successful runs as well as the mean objective function values.

The value of this performance index for a computational algorithm under comparison is given by

$$PI = \frac{1}{N_p} \sum_{i=1}^{N_p} \left( k_1 a_1^i + k_2 a_2^i + k_3 a_3^i \right)$$

where

$$a_1^i = \frac{Sr^i}{Tr^i},$$

$$a_2^i = \begin{cases} 
\frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0 \\
0, & \text{if } Sr^i = 0 
\end{cases},$$

and

$$a_3^i = \frac{Mo_i}{Ao_i},$$

$$i = 1, 2, \ldots, N_p$$

Table II

<table>
<thead>
<tr>
<th>No.</th>
<th>SR (%)</th>
<th>Average Eval</th>
<th>Mean OBJ</th>
<th>Min OBJ</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>14634</td>
<td>0.000932</td>
<td>0.000728</td>
<td>6.22E-05</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
<td>16338.5</td>
<td>20.6174</td>
<td>0.000553</td>
<td>205.125</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>40224</td>
<td>0.010633</td>
<td>0.000693</td>
<td>0.012185</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>50000</td>
<td>6524.176</td>
<td>9.685575</td>
<td>22915.9</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>70.13881</td>
<td>28.85807</td>
<td>0.000069</td>
<td>0.012711</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>30349</td>
<td>0.000960</td>
<td>0.000790</td>
<td>4.16E-05</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>25527.5</td>
<td>0.146521</td>
<td>0.000716</td>
<td>0.315224</td>
</tr>
<tr>
<td>8</td>
<td>75</td>
<td>23200</td>
<td>0.067338</td>
<td>0.000672</td>
<td>0.387977</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>49011</td>
<td>183.8900</td>
<td>0.000830</td>
<td>225.5512</td>
</tr>
<tr>
<td>10</td>
<td>47</td>
<td>34648</td>
<td>0.106840</td>
<td>0.000803</td>
<td>0.116119</td>
</tr>
<tr>
<td>11</td>
<td>99</td>
<td>9977</td>
<td>0.004850</td>
<td>0.000661</td>
<td>0.039058</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>50000</td>
<td>64.29140</td>
<td>21.74521</td>
<td>21.34044</td>
</tr>
<tr>
<td>13</td>
<td>54</td>
<td>39971.5</td>
<td>46.00050</td>
<td>0.000709</td>
<td>49.83928</td>
</tr>
<tr>
<td>14</td>
<td>26</td>
<td>40813.5</td>
<td>2.590243</td>
<td>0.000825</td>
<td>2.89245</td>
</tr>
<tr>
<td>15</td>
<td>57</td>
<td>35652</td>
<td>5.900546</td>
<td>0.000799</td>
<td>7.885655</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>SR (%)</th>
<th>Average Eval</th>
<th>Mean OBJ</th>
<th>Min OBJ</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>14678</td>
<td>0.000929</td>
<td>0.000683</td>
<td>6.07E-05</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
<td>16549</td>
<td>0.000880</td>
<td>0.000280</td>
<td>0.000129</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>40399</td>
<td>0.011728</td>
<td>0.000692</td>
<td>0.012711</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>50000</td>
<td>182.1636</td>
<td>1.023445</td>
<td>522.8666</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>68.66827</td>
<td>24.87396</td>
<td>24.34658</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>29723</td>
<td>0.000959</td>
<td>0.000847</td>
<td>3.15E-05</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
<td>23487</td>
<td>0.101370</td>
<td>0.000488</td>
<td>0.301389</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
<td>25943.5</td>
<td>0.052521</td>
<td>0.004933</td>
<td>0.277426</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>48206</td>
<td>190.6801</td>
<td>0.000622</td>
<td>229.0997</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>21681.5</td>
<td>1.023445</td>
<td>0.000358</td>
<td>6.11E-05</td>
</tr>
<tr>
<td>11</td>
<td>99</td>
<td>9502.5</td>
<td>0.000916</td>
<td>0.000533</td>
<td>8.24E-05</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>32501.5</td>
<td>0.000936</td>
<td>0.000691</td>
<td>6.11E-05</td>
</tr>
<tr>
<td>13</td>
<td>54</td>
<td>32501.5</td>
<td>0.000936</td>
<td>0.000691</td>
<td>6.11E-05</td>
</tr>
<tr>
<td>14</td>
<td>26</td>
<td>17433.5</td>
<td>0.000912</td>
<td>0.000593</td>
<td>7.75E-05</td>
</tr>
<tr>
<td>15</td>
<td>57</td>
<td>26005</td>
<td>0.000950</td>
<td>0.000790</td>
<td>3.96E-05</td>
</tr>
</tbody>
</table>
Table III
COMPARATIVE RESULTS OF PSO-C AND LXPSO-C

<table>
<thead>
<tr>
<th>No.</th>
<th>PSO-C</th>
<th>SX (%)</th>
<th>Average Eval</th>
<th>Mean OBJ</th>
<th>Min OBJ</th>
<th>SD</th>
<th>SX (%)</th>
<th>Average Eval</th>
<th>Mean OBJ</th>
<th>Min OBJ</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>6908.5</td>
<td>0.000902</td>
<td>0.00533</td>
<td>8.86E-05</td>
<td></td>
<td>100</td>
<td>6917</td>
<td>0.000912</td>
<td>0.00706</td>
<td>6.82E-05</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>6202</td>
<td>0.000868</td>
<td>0.00573</td>
<td>9.33E-05</td>
<td></td>
<td>100</td>
<td>6271.5</td>
<td>0.000872</td>
<td>0.00566</td>
<td>0.00107</td>
</tr>
<tr>
<td>3</td>
<td>52</td>
<td>29722.5</td>
<td>0.009344</td>
<td>0.00656</td>
<td>0.013928</td>
<td></td>
<td>59</td>
<td>29169</td>
<td>0.007210</td>
<td>6.42E-06</td>
<td>0.10081</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>50000</td>
<td>34.76020</td>
<td>2.91574</td>
<td>28.6507</td>
<td></td>
<td>0</td>
<td>50000</td>
<td>34.5333</td>
<td>0.140955</td>
<td>30.30104</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>50000</td>
<td>32.60200</td>
<td>13.92940</td>
<td>9.25124</td>
<td></td>
<td>0</td>
<td>50000</td>
<td>29.45223</td>
<td>13.92942</td>
<td>7.71155</td>
</tr>
<tr>
<td>6</td>
<td>62</td>
<td>27468.5</td>
<td>0.472819</td>
<td>0.000784</td>
<td>0.633273</td>
<td></td>
<td>100</td>
<td>15752.5</td>
<td>0.000951</td>
<td>0.000844</td>
<td>3.80E-05</td>
</tr>
<tr>
<td>7</td>
<td>94</td>
<td>8338</td>
<td>0.007081</td>
<td>0.000612</td>
<td>0.024425</td>
<td></td>
<td>97</td>
<td>8114</td>
<td>0.002970</td>
<td>0.000686</td>
<td>0.014393</td>
</tr>
<tr>
<td>8</td>
<td>83</td>
<td>13961.5</td>
<td>0.004699</td>
<td>0.001612</td>
<td>0.022203</td>
<td></td>
<td>84</td>
<td>13720.5</td>
<td>0.002691</td>
<td>0.000242</td>
<td>0.004372</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>8668</td>
<td>0.000910</td>
<td>0.000693</td>
<td>7.65E-05</td>
<td></td>
<td>100</td>
<td>8595.5</td>
<td>0.000913</td>
<td>0.00057</td>
<td>7.43E-05</td>
</tr>
<tr>
<td>10</td>
<td>35</td>
<td>35289.5</td>
<td>0.148096</td>
<td>0.000697</td>
<td>0.138301</td>
<td></td>
<td>100</td>
<td>11878</td>
<td>0.000721</td>
<td>1.74E-05</td>
<td>0.00285</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>4543.5</td>
<td>0.000910</td>
<td>0.000693</td>
<td>7.65E-05</td>
<td></td>
<td>100</td>
<td>4513.5</td>
<td>0.000903</td>
<td>0.000617</td>
<td>8.03E-05</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>50000</td>
<td>1.459300</td>
<td>0.177401</td>
<td>1.03659</td>
<td></td>
<td>0</td>
<td>50000</td>
<td>1.641491</td>
<td>0.166948</td>
<td>1.269332</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>14785</td>
<td>0.000904</td>
<td>6.63E-04</td>
<td>6.92E-05</td>
<td></td>
<td>100</td>
<td>14797</td>
<td>0.000912</td>
<td>0.000705</td>
<td>6.85E-05</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>6224</td>
<td>0.000900</td>
<td>8.33E-05</td>
<td>8.33E-05</td>
<td></td>
<td>100</td>
<td>6274</td>
<td>0.000891</td>
<td>0.000564</td>
<td>9.56E-05</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>13805</td>
<td>0.000925</td>
<td>0.000125</td>
<td>0.000111</td>
<td></td>
<td>100</td>
<td>13700</td>
<td>0.000931</td>
<td>0.000386</td>
<td>7.78E-05</td>
</tr>
</tbody>
</table>

$SR^i$ = Number of successful runs of $i^{th}$ problem,  
$Tr^i$ = Total number of runs of $i^{th}$ problem,  
$A^i$ = Minimum of average number of function evaluations used by all algorithms in obtaining the solutions (satisfying (12)) of $i^{th}$ problem,  
$A^i$ = Average number of function evaluations used by an algorithm in obtaining the solutions (satisfying (12)) for the $i^{th}$ problem,  
$Mo^i$ = Minimum of mean objective function value obtained by all the algorithms in obtaining the solution of $i^{th}$ problem,  
$Mo^i$ = Mean objective function value obtained by an algorithm in obtaining the solution of $i^{th}$ problem,  
$N_p$ = Total number of problems analyzed.  

$k_1, k_2$ and $k_3$ ($k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$) are the weights assigned to percentage of success, average number of function evaluations of successful runs, and mean objective function value respectively. PI is a function of $k_1, k_2$ and $k_3$. Since $k_1 + k_2 + k_3 = 1$, one of $k_i$ could be eliminated to reduce the number of dependent variables from the expression of PI. It is difficult to analyze the 3-D plots of PI for PSOs and LXPSOs. So, we adopt the same methodology as given in [14], i.e., two out of three weights in the expression are set to equal value. PI now becomes a function of only one variable. The resultant cases are as follows:

(i) $k_1 = W$, $k_2 = k_3 = \frac{(1-W)}{2}$; $0 \leq W \leq 1$

(ii) $k_2 = W$, $k_1 = k_3 = \frac{(1-W)}{2}$; $0 \leq W \leq 1$

(iii) $k_3 = W$, $k_2 = k_1 = \frac{(1-W)}{2}$; $0 \leq W \leq 1$

The graphs corresponding to each of the cases (i), (ii) and (iii) are shown in figure 4 (a), (b) and (c) respectively. In these figures the horizontal axis represents the weight $W$ and the vertical axis represents the performance index PI.

![Figure 4(a)](image-url)

In case (i), the average number of function evaluations of successful runs and the mean objective function value are given equal weights. PI values for all four algorithms are superimposed in the Figure 4(a) for comparison and to get a ranking of the performance of the four algorithms. It is observed that for LXPSO-C the value of PI is higher than all the remaining three PSOs. The order of performance of the remaining PSOs is PSO-C > LXPSO-W > PSO-W.
LXPSOs are better than their respective classical versions. However, in this paper, the particles that participated in the crossover are selected randomly. In future work, a suitable selection scheme will be designed and evaluated.

Laplace crossover dispenses off-springs proportional to the spread of parents but for maintaining higher diversity one can choose a distribution such that corresponding crossover dispenses off-springs inversely proportional to the parents. In this paper only worst particle is modified but more than one particle based on some selection scheme can be modified. Also in every iteration, we are performing crossover but further research can be done for a suitable crossover probability. In this paper we have just introduced the idea of implementing a new crossover to PSO. It is expected that in future, the Laplace Crossover operator shows a great potential for further research in PSO.

VI. CONCLUSION AND FUTURE WORK

In this paper a real coded crossover operator, called the Laplace Crossover operator (LX) for PSO, is introduced. LX is a parent centric crossover operator and found to exhibit self-adaptive behavior. By implementing LX in PSO with time varying inertia weight and PSO with constriction factor two new PSOs namely LXPSO-W and LXPSO-C are proposed. In every iteration, the worst particle is replaced by an offspring generated by two randomly selected parents (particles) from the current swarm. The experiments are performed on a set of 15 benchmark problems available in the literature. The user can control the number of dimensions in these problems. Through statistical and graphical analyses of results of PSOs and LXPSOs it is found that LXPSOs perform better than their respective classical versions.

REFERENCES