Dynamic Swarm Dispersion in Particle Swarm Optimization for Mining Unsearched Area in Solution Space (DSDPSO)

Anvar Bahrampour
bahrampour@iausdj.ac.ir
Department of Computer Engineering, Islamic Azad University, Sanandaj Branch, Sanandaj, Iran

Omid Mohamad Nezami
mnezami@iaubijar.ac.ir
Department of Computer Engineering, Islamic Azad University, Bijar Branch, Bijar, Iran

Abstract
Premature convergence in particle swarm optimization (PSO) algorithm usually leads to gaining local optimum and preventing from surveying those regions of solution space which have optimal points in. In this paper, by applying special mechanisms, suitable regions were detected and then swarm was guided to them by dispersing part of particles on proper times. This process is called dynamic swarm dispersion in PSO (DSDPSO) algorithm. In order to specify the proper times and to rein the evolutionary process alternating between exploring and exploiting behaviors, we used a diversity measuring approach and implemented the dispersion mechanism. To promote the performance of DSDPSO algorithm, three different policies including particle relocation, velocity settings of dispersed particles and parameters setting were applied. We compared the promoted algorithm with similar new approaches and according to the numerical results, the proposed algorithm outperformed the basic GPSO, LPSO, DMS-PSO, CLPSO and APSO in most of the 12 standard benchmark problems with different properties taken in this study.

Keywords
Evolutionary algorithm, particle swarm optimization algorithm, population diversity, premature convergence, exploring and exploiting.

1 Introduction
The particle swarm optimization (PSO) invented by Eberhart and Kennedy (1995a,b) is applied to the concept of social interaction for problem solving (Soleymani et al., 2007). Each particle, denoted by $X_i$, represents a point in search space or a solution of the problem. The PSO algorithm iteratively modifies the point and the velocity of each particle as it searches for the optimal solution based on Equation 1

$$V_{id} = \omega \ast V_{id} + c_1r_1(p_{id} - x_{id}) + c_2r_2(p_{gd} - x_{id})$$

$$X_{id} = X_{id} + V_{id}$$

(1)

Where $V_i$ in the first equation is the velocity of the $i$th particle. The first part of Equation 1 ($\omega \ast V_{id}$) is the inertia of the previous velocity in which $\omega$ is predefined by the user. The second part ($c_1r_1(p_{id} - x_{id})$) represents the cognition of the particle
which shows personal thinking of the particle and the third part \((c_2 r_2 (p_{gd} - x_{id}))\) is a social component. In this equation, \(c_1\) and \(c_2\) are acceleration constants. They represent weights of the stochastic acceleration terms which pull each particle toward the personal and global best positions. The constants \(r_1\) and \(r_2\) are the uniformly generated random numbers in interval \((0, 1]\). Although PSO is simple, it is a powerful search technique which has been reported to have a satisfactory performance according to many studies (Cheng and Shi, 2011).

The convergence rate of particles in PSO is good through the fast information flowing among particles, so its diversity decreases rapidly in the successive iterations and leads to a suboptimal solution. In this case, it is said that an evolutionary process has been trapped in a local optimum or premature convergence of the process has been occurred.

The standard PSO algorithm can easily get trapped in a local optimum while solving complex multimodal problems. Such deficiencies have restricted the wider applications of the PSO (Zhan et al., 2009; Liang et al., 2006; Li and Engelbrecht, 2007). There are several reasons why such problem arises. One of the most important reasons is decreasing diversity of the population. A number of variants of PSO algorithm have been proposed to overcome the problem of diversity loss. One of the common methods to increase diversity is mutation. Mutation leads to the improvement of exploration abilities, which can be applied to different elements of a particle swarm. The effect of mutation depends on which elements of the swarm are mutated (Secrest and Lamont, 2003). Velocity vector mutation is equivalent to particles position vector mutation provided that the same mutation operator is considered.

Secrest and Lamont (2003) proposed a negative feedback mechanism into particle swarm optimization and developed an adaptive PSO as well. This mechanism takes advantage of the swarm-diversity to control the tuning of the inertia weight (PSO-DCIW), which in turn can adjust the swarm-diversity adaptively and make a contribution to a successful global search. There are other methods including Gaussian Mutation (Wei et al., 2002; Higashi and Iba, 2003; Secrest and Lamont, 2003; Sriyanyong, 2008; Krohling, 2005; Cauchy (Krohling, 2005; Stacey et al., 2003), and Chaos Mutation (Dong et al., 2008; Yang et al., 2009; Yue-Lin et al., 2008) which measure diversity and apply mutation in particles positions to improve the performance of the algorithm.

There are other ways to introduce diversity and to control its degree. Zhan et al. (2009) proposed an algorithm named APSO to do so. In this method, they utilized automatic control of algorithmic parameters. A learning strategy whereby all other particles’ historical best information was used to update a particle’s velocity was suggested by Liang et al. (2006) and called CLPSO. Riget and Vesterstrom (2002) proposed an algorithm named ARPSO in which if diversity was above the predefined threshold \(d_{\text{high}}\), particles attracted each other; and if it was below \(d_{\text{low}}\), particles repelled each other until they met the required high diversity \(d_{\text{high}}\). Repulsion to keep particles away from the optimum was first proposed by Parsopoulos and Vrahatis (2004). Lovbjerg and Krink (2002) made dispersion among those particles which were too close to each other; and Blackwell and Bentley (2002) reduced the attraction of the swarm centers in order to prevent the particles from tight clustering in one region of the search space and to escape from local optimum. A dynamic multi-swarm particle swarm optimizer (DMS-PSO) was proposed by Liang and Suganthan (2005). In this method, the whole population was divided into many small swarms. These swarms were frequently regrouped by using various regrouping schedules and information was exchanged among the swarms.
The diversity level of the swarm during the evolutionary process was measured by [Mohamad Nezami and Bahrampour, 2013]. In their study, once the diversity of the population drops down to the predefined threshold \( d \), the system of generating diversified artificial particles (DAP system) is activated and starts to replace some of the particles of swarm which have relatively bad fitness by generated artificial particles (DAP particles) which have high diversity and fairly good fitness. [Bahrampour and Mohamad Nezami, 2013] and [Mohamad Nezami et al., 2013] investigated more profoundly and promoted this basic idea by three definitions and concepts including idle particles, relocation or dispersion terms and precise search in new regions of the search space by artificial particles. They proposed a mechanism to guide the swarm based on diversity by using a diversifying process in order to detect suitable positions of the search space (points with fairly good fitness and good distance from current distribution of the swarm particles) to disperse or relocate some of the existing idle particles (those particles that there has been no change in their personal best positions for long time) in the hope of increasing diversity level of the swarm and escaping from local optimum by detecting better area of the search space. The algorithm proposed by [Mohamad Nezami and Bahrampour, 2013] was improved by defining new velocity equation for artificial particles which was used in a limited duration after each replacement to search more carefully and precisely in new regions of the search space [Mohamad Nezami et al., 2013]. In this paper, all the previous works are engaged and a comprehensive study is conducted on the behavior of PSO algorithm with the aforementioned ideas. In addition, we prove the policies and parameters used for designing dynamic swarm dispersion particle swarm optimization (DSDPSO) algorithm.

The article is organized as follows. In section 2, diversity and measuring are defined. The DSDPSO algorithm is described in section 3. Experimental results are discussed in section 4. Finally, conclusions are mentioned in section 5.

2 Diversity Definition and Measuring

Population diversity is a way to monitor the degree of convergence or divergence in PSO search process [Cheng and Shi, 2011]. There are several measures to detect diversity level of the population. [Shi and Eberhart, 2008, 2009] and [Zhan et al., 2010] gave three definitions for PSO population diversity measurements including position diversity, velocity diversity, and cognitive diversity. [Cheng and Shi, 2011] gave new definition for population diversity measurement, called \( L_1 \) norm, based on both element-wise and dimension-wise diversities. They showed that useful information on search process of an optimization algorithm could be obtained by using dimension-wise definition in \( L_1 \) norm variant. Therefore, we apply \( L_1 \) norm of position diversity measurement in this paper. Let \( m \) be the number of particles and \( n \) the number of dimensions. Dimension-wise definition in \( L_1 \) norm is defined as follows (Equation 2).

\[
\bar{x}_i = \frac{1}{m} \sum_{i=1}^{m} x_{ij} \\
D^p = \frac{1}{m} \sum_{i=1}^{m} |x_{ij} - \bar{x}_j| \\
D^p = \frac{1}{n} \sum_{i=1}^{n} D^p_j
\]

Where vector \( \bar{x} \) is mean of particle position on each dimension, \( D^p \) is particle position diversity vector based on \( L_1 \) norm, and \( D^p \) is the whole population diversity.
Chang et al. (2010) introduced other approaches to measure population diversity in evolutionary computation including Hamming distance, Euclidean distance, information Entropy, and so on. In this paper, we apply Euclidean distance in particles selection process to disperse them in dispersion mechanism. The Euclidean distance, defined as Equation 3, measures the distance between two particles $X$ and $Y$.

$$D(X, Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

### 3 Dynamic Swarm Dispersion Particle Swarm Optimization

The problem of premature convergence in particle swarm optimization (PSO) algorithm often causes that the search process gets trapped in a local optimum. This problem usually occurs when the diversity of the swarm decreases and the swarm cannot escape from a local optimum. In this article, we periodically disperse some of the swarm’s particles to new suitable positions with fairly good fitness in the search space and with relatively far distance from convergence point. These new points in the search space are recognized based on the history of the search process in order to enhance the diversity of the swarm and to promote the exploration ability of the algorithm. In other words, the search process should periodically select some of the converged particles in current swarm and relocate them to new different points of the search space in order to probe new regions of the search space. Both selection process and the process of detecting new target positions of the relocated particles act based on the history of the search process up to dispersion time. In this approach, we do not change the previous personal best positions of dispersed particles in dispersion stage in order to use the result of the efforts made by relocated particles previously. Therefore, this is a relocation of the selected particles to new suitable positions rather than the replacement of them with new generated ones. Consequently, the diversity level of the swarm will be increased up to a certain degree. The evolutionary process will consistently reduce the diversity level again, and the dispersion process should be repeated periodically.

In PSO algorithm, the speed of convergence is very high, so the swarm dispersion process should be repeated frequently. On the other hand, repetition of this process is relatively time-consuming and exploitation ability of the algorithm will be decreased by high frequency of swarm dispersion, too. Therefore, we introduce a new parameter $T$ to define the period of the dispersion system call in new search process. Another new parameter in DSDPSO algorithm which defines the amount of the swarm and should be relocated in each dispersion call is dispersion rate $R$.

The architecture of a DSDPSO is illustrated in Figure 1. In DSDPSO, we partially disperse swarm to prevent from premature convergence and to extend those search spaces which are probably unexplored in the simple PSO algorithm. In Figure 1, there are two different phases in process of dynamic swarm dispersion in particle swarm optimization algorithm. The first phase, which has been illustrated on the left side of Figure 1 is a simple PSO plus one last step for updating an external archive to be used in dispersion mechanism. In DSDPSO, we normally execute phase one in every iteration. In the second phase, shown on the right side of Figure 1, the selected particles of the swarm are dispersed to the new positions in search space in each $T$ period. This phase will enhance the swarm diversity by relocating $R\%$ of the swarm in search space in order to escape from local optimum.
Dynamic Swarm Dispersion in Particle Swarm Optimization for Mining Unsearched Area

Figure 1: Architecture of DSDPSO

Figure 2 shows the process of dynamic swarm dispersion particle swarm optimization (DSDPSO) algorithm. The steps of this process are identical to the steps of the standard PSO except for steps 5 and 6. In order to detect target positions of the selected particles for dispersion, we apply the information collected from previous generations of the PSO process using the previous global best particles to determine good points in the search space. To do so, we develop an external archive to store the global best particles of previous generations as good positions in the search space which have been visited in the hope of finding better points in the regions in which these stored particles are located. In step 4, we update the external archive if necessary; since there is no necessity for the external archive to be updated in every iteration (the process updates external archive only when notable improvement occurs in the fitness of the global best particle).

When dispersion system is called in step 5, last dispersion has taken place $T$ generations ago. At this time, dispersion process selects $R\%$ of the swarm’s particles, determines the same number of target positions, and relocates them to new positions. Dispersion process will increase the swarm diversity by relocating selected particles to new potent positions. The process of determining new target positions and selecting policy of swarm’s particles for relocation will be described in the following sections. The final step of dispersion process is to reset the velocity of dispersed particles to zero.
because we need each dispersed particle for a very careful search in order to find better solutions in the vicinity of new location. In this paper, we found out that the previous nonzero velocities of dispersed particles are probably the cause of rapid skipping from new region and subsequently having some areas remain unsearched in the search space. This idea will be verified in section 3.2.2.

In step 6, we use two different velocity equations, Equation 1 and Equation ?? to update velocity of the swarm particles. The following velocity equation is used only for those dispersed particles generated in the latest dispersion process call. Since we want to search new regions more carefully by relocated particles, we use the minimum inertia weight. Moreover, since relocated particles are located in the regions with probably far distance from the converged particles, it is likely to have a big component of velocity to the global best particle in velocity equation. Thus, we use a random coefficient in interval (0, 1) in new velocity equation in order to relocate the particles slowly attracted to the converged swarm. This idea is verified in section 3.2.3.

\[
V_{id} = r_0(\omega_{\text{min}} \ast V_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}))
\]

\[
X_{id} = X_{id} + V_{id}
\]

Figure 2: Steps of DSDPSO algorithm

To illustrate the impact of dispersion mechanism on diversity level of the swarm in DSDPSO, we use a 2-D Rastringin function (f7 in Table 1) with the swarm of 30 particles
and the dispersion rate of 45%. The swarm distribution in different running phases of this experiment are shown in Figure 3. The stochastic initialization of particles in the first iteration is illustrated in Figure 3(a). Then, the learning mechanism of the algorithm pulls particles toward the optimal region in the 30th iteration, as illustrated in Figure 3(b). In the 30th iteration, DS-DPSO algorithm relocates some particles of the current swarm to increase the swarm diversity based on dispersion mechanism as shown in Figure 3(c). In this stage, the new swarm is named dispersed swarm. Figure 3(e) and Figure 3(f) show behaviors of the swarm in the 60th iteration same as Figure 3(b) and Figure 3(c) respectively. Figure 3(c) and 3(f) illustrate the achievement of dispersion mechanism for dispersing particles throughout the search space properly. It is noteworthy that in dispersion mechanism despite the increasing diversity of the swarm, the fitness level of new positions is not worse than the particles fitness before relocating. So, in DS-DPSO algorithm, target positions can be generated with greater diversity and better fitness as well. Figure 3(d) and 3(g) substantiate this claim.

Finally, Figure 4 represents the diversity curves of standard global PSO and DS-DPSO. It shows how diversity level of the swarm changes in GPSO and DS-DPSO in 100 iterations.

Figure 3: (a) Stochastic distribution of swarm in generation 1. (b) Swarm distribution in iteration 30. (c) Swarm distribution in iteration 30 after dispersion call. (d) Fitness diagrams of swarm of b (converged pop) and c (dispersed pop) states. (e) Swarm distribution in iteration 60. (f) Swarm distribution in iteration 60 after dispersion call. (g). Fitness diagrams of swarm of e (converged pop) and f (dispersed pop) states.
3.1 Target Positions of Selected Particles

In this section, we describe a mechanism for determining target positions of the selected particles to disperse over the search space. In this paper, we have established an external archive with 100 particles and initialized it with random particles. Firstly, we gather those particles having the best fitness in the first 100 generations of the PSO process and use them to replace the particles in the external archive having bad fitness. Then, we should establish a replacement policy in order to gather effective particles in external archive. These particles should have both good fitness and considerable distance from the center of current distribution of the external archive particles to avoid the convergence of external archive. In this study, after the first 100 generations, replacement is applied only when the fitness of global best particle changes remarkably. In that case, one of the particles with low diversity will be removed by the new global best particle of the current swarm. To detect particles with low diversity and remove them from external archive, we use Euclidean distance described in section 2 and measure distance of each particle from the mean of the external archive particles.

Figure 5 illustrates the mechanism of determining target positions. To determine the new good positions for relocating converged particles of the swarm from information of the external archive, two new particles of \( x_{\text{max}} \) and \( x_{\text{min}} \) (vector of maximum or minimum value in each dimension) are added to external archive for mutation purpose. Then, a Roulette wheel is created to weigh each particle of the external archive based on its fitness and distance from the center of the swarm. In order to detect new target position in the search space, we should determine the value of each dimension of target position. Therefore, for each dimension value, one Roulette wheel selection can select a particle of the external archive, and the value stored in the same dimension of the selected particle will be used to generate the same dimension value of new target position. After selecting the value of each dimension, we probably have a new suitable position for dispersion process. However, we do not use this point as good position for dispersion this time. We collect all the generated points (100 points in this study) in one matting pool and add the external archive particles as good points in the search space.
to the pool, too. Then, operators such as genetic crossover and mutation are applied to produce new points with probably good fitness and good diversity. Afterwards, a numbers of best points (45% of the swarm in this research) are selected based on their fitness and distance from the center of the current swarm distribution and returned to dispersion process to relocate randomly the same numbers of selected particles of the swarm to these new positions. This process will remarkably increase diversity of the swarm and help to escape from local optimum trap. Figure 6 illustrates this process.

Figure 5: Steps of detecting target positions in DSDPSO algorithm

Figure 6: Process of determining target positions

Ex=Ex + MaxParticle; // MaxParticle is a vector of xMax
Ex=Ex + MinParticle; // MinParticle is a vector of xMin
Roulette = Roulettewheel(Ex); // External Archive
// Create a Roulette weight (p) (1+fitness)/sum(fitness) + distance(p)/sum(distance)
For each new Point p do
  For each dimension d do
    Particle = select (Roulette);
    Pool (p, d) = Ex (Particle, d);
  End
End
Pool = Pool + External Archive;
Crossover selected Points from Pool and does Mutation operator
Select good points and return to Dispersion process
3.2 Performing Policies

As mentioned in the previous section, to promote performance of the DSDPSO algorithm, we have to properly perform several policies. In this section, we study three policies including Relocation policy, initial velocity of dispersed particles, and velocity equation of dispersed particles. These policies will be elaborated in the following three subsections. In the first subsection, we explain about different methods for selecting particles of current population for relocation and then compare them together. In the second subsection, initial velocity of dispersed particles will be explained. Finally, in the last subsection, we test different policies to control behavior of the dispersed particles and prove their abilities. In order to comparison of different approaches in each policy, we have used a collection of 10 standard benchmark problems. Mathematical models of the problems along with the true optimal value are given in Table 1.

Table 1: Benchmark functions used in our experimental study

| Function                      | Function Definition                                                                 | Interval         | Optimum |
|-------------------------------|------------------------------------------------------------------------------------|------------------|---------|
| Sphere Function               | $f_1(x) = \sum_{i=0}^{n-1} x_i^2$                                                  | [-100,100]       | 0       |
| Schwefel Function 1.2         | $f_2(x) = \sum_{i=0}^{n-1} (\sum_{j=0}^i x_j)^2$                                   | [-100,100]       | 0       |
| Schwefel’s Problem 1.2 with Noise | $f_3(x) = (\sum_{i=0}^{n-1} (\sum_{j=0}^i x_j)^2) \times (1 + 0.4(|N(0, 1)|))$ | [-100,100]       | 0       |
| Noisy Function                | $f_4(x) = (\sum_{i=0}^{n-1} (i + 1) x_i) + \text{rand}[0, 1]$                      | [-128,128]       | 0       |
| Rosenbrock Function           | $f_5(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$                   | [-30,30]         | 0       |
| Schwefel Function             | $f_6(x) = 418.98920n - \sum_{i=0}^{n-1} x_i \sin(\sqrt{|x_i|})$                   | [-500,500]       | 0       |
| Rastrigin Function            | $f_7(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$                        | [-5.12,5.12]     | 0       |
| $f_8(x) = \sum_{i=1}^{n} [y_i^2 - 10 \cos(2\pi y_i) + 10]$                      | [-5.12,5.12]     | 0       |
| Noncontinuous Rastrigin Function | $g_i = \frac{x_i + \text{round}(2x_i)}{2}$ if $|x_i| < \frac{1}{2}$, if $|x_i| \geq \frac{1}{2}$ | [-32,32]         | 0       |
| Schaffer’s Function           | $f_9(x) = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}} \sin^2(50(\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}} + 1.0)$ | [-32.76,32.76]   | 0       |
| Griewank Function             | $f_{10}(x) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 + \sum_{i=0}^{n-1} \cos \left( \frac{x_i}{\sqrt{i+1}} \right) + 1$ | [-600,600]       | 0       |
| Rotated Ackley Function       | $f_{11}(x) = 20 + e - 20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} z_i^2}) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi z_i)\right)$ | [-32,32]         | 0       |
| Rotated Rastrigin’s Function  | $f_{12}(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi z_i) + 10)$                     | [-5.12,5.12]     | 0       |
3.2.1 Particles Relocation Policy

Selecting particles and relocating them to new positions, which was completely explained in section 3.1, is performed by three approaches. In the first approach, particles with low fitness are selected for dispersion. Since probability of particles with low fitness is too low to find the optimal points of the search space, this approach is applied. It is supposed that regions around these particles are not probably suitable enough to find global optimum. In the next approach, idle particles will be relocated to new points in the search space. Idle particle is a particle that there is no change in its personal best position for long period of time; therefore, this particle could not probably find better locations.

Each of these two approaches might have some shortcomings; in the first approach, we disperse particles with low fitness, even though these particles may find better locations in subsequent generations and their fitness would be improved alternatively. In the second approach, the particles are sometimes relocated because of there is no change in their local best positions for a while, regardless to the fact that they may have high fitness. To avoid such problems, we introduce the third approach. This approach is a combination of the two previous ones. In the hybrid approach, particles with relatively low fitness and with no change in their local best positions for a long term are dispersed. Results of these approaches are indicated in Table 2. According to the results, the hybrid approach leads to the best policy for dispersion.

Table 2: The results of particles relocation approaches

| Relocation | Low Fitness | Idle Particle | Hybrid    |
|------------|-------------|---------------|-----------|
| \( f_1 \)  | 1.91e-4     | 9.68e-6       | 6.64e-6   |
| \( f_2 \)  | 2.03e-5     | 2.62e-7       | 5.46e-8   |
| \( f_3 \)  | 2.77e-2     | 2.48e-2       | 2.16e-2   |
| \( f_4 \)  | 2.31e+3     | 1.01e+3       | 1.12e+3   |
| \( f_5 \)  | 4.73e+1     | 2.92e+1       | 2.60e+1   |
| \( f_6 \)  | 8.67e-3     | 9.02e-4       | 5.04e-4   |
| \( f_7 \)  | 6.31e-3     | 3.95e-3       | 8.62e-4   |
| \( f_8 \)  | 1.23e+1     | 4.36e+0       | 3.70e+0   |
| \( f_9 \)  | 1.39e+1     | 7.14e+0       | 9.25e+0   |
| \( f_{10} \)| 1.42e+0     | 1.19e+0       | 1.08e+0   |

3.2.2 Velocity Settings of Dispersed Particles

When particles are dispersed to new positions in the search space, some of their features such as previous velocities (initial velocity after relocation), the local best positions and inertia weight are unknown. For previous velocity of a dispersed particle, we have three options including final velocity of the particle just before relocation, a random velocity, and a zero velocity. However, it is not reasonable to use previous velocity of the particle, final velocity just before relocation, in computing new velocity of the particle in new position. Random velocity causes the dispersed particles to scat from the detected new region of the search space as the new positions are intentionally selected based on the probability of finding optimal point near new target positions. In order to avoid the latter problem and search carefully in new region, we test zero initial velocity in dispersed particles after dispersion. These three approaches are evaluated by the mentioned test functions and results are illustrated in Table 3. In this experiment, the
particles with low fitness are selected for dispersion. According to the results, setting the initial velocities to zero in dispersion stage is the best approach.

Table 3: Results of the three approaches for calculating rudimentary velocities

| Initial Velocity | Random Velocity | Zero Velocity | Previous Velocity |
|------------------|-----------------|--------------|------------------|
| $f_1$            | 1.80e−04        | 5.48e−08     | 1.82e−07         |
| $f_2$            | 1.31e−05        | 2.77e−11     | 5.07e−11         |
| $f_3$            | 2.65e−02        | 6.94e−03     | 8.60e−03         |
| $f_4$            | 2.96e+03        | 1.16e+03     | 1.23e+03         |
| $f_5$            | 3.38e+01        | 2.46e+01     | 2.47e+01         |
| $f_6$            | 1.12e−02        | 2.20e−06     | 8.07e−06         |
| $f_7$            | 3.30e−03        | 6.16e−04     | 2.46e−03         |
| $f_8$            | 1.33e+01        | 1.62e+00     | 3.81e−08         |
| $f_9$            | 1.32e+01        | 2.63e+00     | 1.48e+00         |
| $f_{10}$         | 1.40e+00        | 2.84e−01     | 4.03e−01         |

3.2.3 Behavior of Dispersed Particles

As mentioned in Section 3, since we want to search more carefully new regions of the search space by relocated particles, the dispersed particles should slowly move toward the global and local best particles. Therefore, big components of the velocity equation (Equation 1) violate this aim, so we use the minimum inertia weight for dispersed particles for a period $T$ after dispersion. Moreover, since the relocated particles are located in the regions with probably far distance from the converged particles, it is likely to have a big component of velocity to the global and local best particles in velocity equation. Thus, we use a random coefficient in interval $(0, 1)$ in new velocity equation (Equation 4) in order to relocate particles slowly attracted to the converged swarm. Equation 4 is applied to update velocities of the dispersed particles for $T$ iterations after dispersion stage.

Table 4 shows that the new equation (Equation 4) outperforms the standard velocity equation (Equation 1) and low value of inertia weight in finding optimal particles. In this experiment, particles with low fitness are selected for dispersion, and the dispersed particles restart search process by initial zero velocity.

Table 4: Results of the three approaches for calculating velocities

| Velocity | PSO Eq. | PSO Eq. with Low inertia | Proposed Eq. 4 |
|----------|---------|--------------------------|----------------|
| $f_1$    | 2.48e−04| 4.18e−06                 | 5.03e−10       |
| $f_2$    | 5.78e−05| 4.73e−10                 | 4.84e−15       |
| $f_3$    | 2.89e−02| 1.44e−02                 | 1.83e−03       |
| $f_4$    | 2.58e+03| 2.43e+03                 | 2.29e+03       |
| $f_5$    | 2.72e+01| 2.52e+01                 | 1.77e+01       |
| $f_6$    | 3.92e−03| 7.27e−05                 | 3.72e−09       |
| $f_7$    | 3.14e−03| 4.06e−03                 | 1.13e−14       |
| $f_8$    | 1.13e+01| 8.27e−01                 | 2.16e+00       |
| $f_9$    | 1.34e+01| 4.19e+00                 | 1.51e+00       |
| $f_{10}$ | 1.57e+00| 1.06e+00                 | 3.29e−01       |
3.3 Parameter Setting

In order to test the effectiveness of the dispersion mechanism, a set of experimental tests were conducted to setup parameters, introduced in section 3.1, including period $T$, dispersion rate $R$, and the size of the external archive. All the experiences in this subsection have been achieved under the same conditions. To setup parameters in high stability, average of 20 runs was applied for each one. Ten standard problems with different properties were used to test the parameter values. First, the experiment was designed to setup parameter $T$. In these experiments, values of $T$ in terms of generation were set to 10, 30, 50, and 70. The final results of these experiments are shown in Table 5. As we expected the lower value of $T$ led to search better fitness in the search space. Secondly, the dispersion rate $R$ was experimented by four different values of 15, 30, 45, and 60 percentage of the swarm. Table 6 shows the results of these experiments, the best result reached by dispersion of 45% of swarm. Table 7 illustrates the results of the experiment by different external archive size. Under the condition of these experiments, the external archive with 100 particles is the best setting for this parameter.

| Period $T$ | 15% | 30% | 45% | 60% |
|------------|-----|-----|-----|-----|
| $f_1$      | $4.61e-10$ | $4.05e-14$ | $2.02e-13$ | $2.15e-06$ |
| $f_2$      | $1.23e-14$ | $1.45e-22$ | $1.11e-24$ | $6.63e-18$ |
| $f_3$      | $1.31e-03$ | $1.52e-03$ | $1.79e-03$ | $3.39e-03$ |
| $f_4$      | $1.95e+03$ | $2.16e+03$ | $2.36e+03$ | $3.20e+03$ |
| $f_5$      | $1.96e+01$ | $2.34e+01$ | $2.31e+01$ | $2.97e+01$ |
| $f_6$      | $2.01e-09$ | $1.87e-13$ | $5.13e-13$ | $2.10e-04$ |
| $f_7$      | $5.28e-03$ | $1.23e-02$ | $1.98e-02$ | $8.35e-03$ |
| $f_8$      | $1.15e-13$ | $8.88e-17$ | $6.22e-16$ | $1.14e+00$ |
| $f_9$      | $5.00e-02$ | $1.17e+00$ | $6.95e+00$ | $1.38e+01$ |
| $f_{10}$   | $7.56e-02$ | $5.04e-01$ | $6.10e-01$ | $1.06e+00$ |
### Table 7: Results of applying different sizes of external archives

| Size of External Archive | 50   | 100   | 150   | 200   |
|--------------------------|------|------|------|------|
| \( f_1 \)               | 1.68e -10 | 7.61e -14 | 1.70e -13 | 5.44e -09 |
| \( f_2 \)               | 5.72e -18 | 1.69e -22 | 3.81e -23 | 2.16e -17 |
| \( f_3 \)               | 2.59e -03 | 1.32e -03 | 8.32e -04 | 1.65e -03 |
| \( f_4 \)               | 2.72e + 03 | 2.21e + 03 | 2.16e + 03 | 2.43e + 03 |
| \( f_5 \)               | 2.73e + 01 | 1.85e + 01 | 2.21e + 01 | 2.23e + 01 |
| \( f_6 \)               | 9.91e - 11 | 2.28e -13 | 2.59e -13 | 8.99e -07 |
| \( f_7 \)               | 8.61e - 03 | 8.47e - 03 | 5.04e - 03 | 1.01e - 02 |
| \( f_8 \)               | 4.51e - 12 | 1.78e - 16 | 0.00e + 00 | 7.87e - 01 |
| \( f_9 \)               | 5.24e + 00 | 2.18e + 00 | 5.05e + 00 | 1.06e + 01 |
| \( f_{10} \)            | 7.57e - 01 | 4.25e - 01 | 3.73e - 01 | 2.67e - 01 |

4 Experimental Setting and Numerical Results

In order to compare some variants of PSO and DSDPSO algorithms, we have used a collection of 12 standard benchmark problems. Mathematical models of the problems along with the true optimal value are given in Table 2. In this problem set, we have unimodal functions such as \( f_1, f_2, f_3 \) and \( f_4 \). \( f_4 \) is a noisy quadric function where a uniformly distributed random variable is in the interval \([0, 1)\). The others are unrotated and rotated multimodal functions (Yao et al., 1999). The entire set of test problems taken for this study is scalable. In other words, the problems can be tested for any number of variables. However, in the present study, we have tested the problems for dimensions 30 and 50.

In order to make a fair comparison between the proposed DSDPSO and other variants of PSO algorithm, we implement standard PSO algorithm in both global star structure and local ring structure named GPSO and LPSO respectively. In addition, we implement APSO, CLPSO and DMS-PSO algorithms proposed in Liang and Suganthan (2005); Liang et al. (2006); Zhan et al. (2009) and compare them with DSDPSO. The same initial population is used for all algorithms. The population size is specified 20 and 50 when there are 30 and 50 variables (dimensions) respectively for all the test problems. A linearly decreasing inertia weight starting at 0.9 and ending at 0.4 is used with the user defined parameters \( c_1 = 2.0 \) and \( c_2 = 2.0 \). For each algorithm, the maximum number of iterations is set to 3000 in the case of having 30 dimensions, and 10000 for dimension 50. For DSDPSO algorithm, a new parameter \( T \) is set to 30 and 50 in case of having 30 and 50 dimensions respectively. The external archive of size 100 and dispersion rate \( R \) of 45% are specified. In DMS-PSO, a group size of 3 and regroup period of 5 are applied. A total of 20 runs are conducted for each experimental setting. The results are given in Table 8 and Table 9 in terms of mean best fitness, standard deviation, and P-Value. Figure 7-11 show performance curves of the DSDPSO in comparison with other variants of PSO for test functions \( f_1, f_5, f_8, f_{11}, \) and \( f_{12} \) by mean fitness of the best particles history found by the swarm in all runs. The numerical results show that the proposed algorithm outperforms other variants of PSO in most of the test cases taken in this study.
Dynamic Swarm Dispersion in Particle Swarm Optimization for Mining Unsearched Area

Figure 7: Performance curves of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for function $f_1$

Figure 8: Performance curves of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for function $f_5$

Figure 9: Performance curves of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for function $f_8$
A. Bahrampour and O. Mohamad Nezami

Figure 10: Performance curves of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for function $f_{11}$

Figure 11: Performance curves of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for function $f_{12}$

5 Conclusion

Evolutionary algorithms (EAs) are the best solutions for solving optimization problems. Although having different abilities to investigate the search space and attain optimal solution, they all behave similarly. One of the ideas to control the behavior of these algorithms is a rein between exploration and exploitation. In such case, we need a good mechanism to enhance the diversity of population in different stages to achieve the unsearched spaces. In order to enhance diversity and survey unsearched spaces, we used a historical approach to search and implemented that on the PSO algorithm. We proposed a mechanism to guide the swarm based on diversity by using a dispersion process in order to detect suitable positions in the search space. This model uses a dispersion mechanism to control the evolutionary process alternating between exploring and exploiting behavior and guide the algorithm, called DSDPSO algorithm, to survey the unsearched spaces. The numerical results showed that the proposed algorithm outperformed the basic GPSO, LPSO, DMS-PSO, CLPSO and APSO in most of the test cases with different properties taken in this study. It is of no doubt that this model can be used on other EAs with a little modification.
Acknowledgements

The authors would like to thank Dr. Ponnuthurai Nagaratnam Suganthan for sending his implementations which helped us to improve the quality of our paper.

References

Bahrampoor, A. and O. Mohamad Nezami (2013). Diversity guided particle swarm optimization (DGPSO) algorithm based on search space awareness particle dispersion. *Proceedings of the 15th International Conference on Artificial Intelligence (ICAI 2013).* Volume 2, pp. 218-224. Las Vegas Nevada, USA.

Blackwell, T. M. and P. Bentley (2002). Dont push me! Collision-avoiding swarms. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002).* Volume 2, pp. 1691-1696. Honolulu, USA.

Chang, P. C., W. H. Huang, and C. J. Ting (2010). Dynamic diversity control in genetic algorithm for mining unsearched solution space in TSP problems. *Expert Systems with Applications 37*(3), 1863-1878.

Cheng, S. and Y. Shi (2011). Diversity control in particle swarm optimization. *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS 2011).* pp. 1-9. Paris, France.

Dong, D., J. Jie, J. Zeng, and M. Wang (2008). Chaos-mutation-based particle swarm optimizer for dynamic environment. *Proceedings of the 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE 2008).* Volume 1, pp. 1032-1037. Xiamen, China.

Eberhart, R., R. Dobbins, and P. Simpson (1996). *Computational Intelligence PC Tools.* Academic Press Professional.

Eberhart, R. and J. Kennedy (1995a). A new optimizer using particle swarm theory. *Proceedings of the 6th International Symposium on Micro Machine and Human Science.* pp. 39-43. Nagoya, Japan.

Higashi, N. and H. Iba (2003). Particle swarm optimization with Gaussian mutation. *Proceedings of the IEEE Swarm Intelligence Symposium.* pp. 72-79. Indianapolis, USA.

Jie, J., J. Zeng, and C. Han (2006). Adaptive particle swarm optimization with feedback control of diversity. *Proceedings of the International Conference on Intelligent Computing (ICIC 2006).* pp. 8192. Kunming, China.

Kennedy, J. and R. Eberhart (1995b). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks (ICNN 1995).* Volume 4, pp. 1942-1948. Perth, Australia.

Krohling, R. A.(2005). Gaussian particle swarm with jumps. *Proceedings of the IEEE Congress on Evolutionary Computation.* pp. 1226-1231. Edinburg, Scotland.

Li, X. and A. P. Engelbrecht (2007). Particle swarm optimization: An introduction and its recent developments. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007).* pp. 3391-3414. London, UK.

Liang, J. J. and P. N. Suganthan (2005). Dynamic multi-swarm particle swarm optimizer with local search. *Proceedings of the IEEE Congress on Evolutionary Computation.* Volume 1, pp. 522528. Edinburgh, Scotland.

Liang, J. J., A. K. Qin, P. N. Suganthan, and S. Baskar (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation 10*(3), 281-295.

Lovbjerg, M. and T. Krink (2002). Extending particle swarm optimisers with self-organized criticality. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002).* Volume 2, pp. 1588-1593. Honolulu, USA.
Mohamad Nezami, O. and A. Bahrampour (2013). Particle swarm optimization algorithm based on diversified artificial particles (PSO-DAP). *Proceedings of the 11th Iranian Conference on Intelligent Systems*. Tehran, Iran.

Mohamad Nezami, O., A. Bahrampour, and P. Jamshidlou (2013). Dynamic diversity enhancement in particle swarm optimization (DDEPSO) algorithm for preventing from premature convergence. *Proceedings of the 17th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*. Seoul, South Korea.

Parsopoulos, K. E. and M. N. Vrahatis (2004). On the computation of all global minimizers thorough particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 8(3), 211-224.

Riget, J. and J. S. Vesterstrom (2002). A Diversity guided particle swarm optimizerThe ARPSO. *Technical Report*. Department of Computer Science, University of Aarhus, Denmark.

Secrest, B. R. and G. B. Lamont (2003). Visualizing particle swarm optimizationGaussian particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003)*. pp. 198-204. Indianapolis, USA.

Shi, Y. and R. Eberhart (2009). Monitoring of particle swarm optimization. *Frontiers of Computer Science in China* 3(1), 31-37. SP Higher Education Press.

Shi, Y. and R. Eberhart (2008). Population diversity of particle swarms. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008)*. pp. 1063-1067. Hong Kong.

Soleymani, S., A. M. Ranjbar, S. Bagheri Shoukari, A. R. Shirani, and N. Sadati (2007). A new approach for bidding strategy of gencos using particle swarm optimization combined with simulated annealing method. *Iranian Journal of Science & Technology, Transaction B, Engineering*. Volume 31, No. B3, pp. 303-315. Shiraz University, Iran.

Sriyanyong, P.(2008). Solving economic dispatch using Particle Swarm Optimization combined with Gaussian mutation. *Proceedings of the 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2008)*. Volume 2, pp. 885-888. Krabi, Thailand.

Stacey, A., M. Jancic, and I. Grundy (2003). Particle swarm optimization with mutation. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2003)*. Volume 2, pp. 1425-1430. Canberra, Australia.

Wei, C., Z. He, Y. Zhang, and W. Pei (2002). Swarm directions embedded in fast evolutionary programming. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*. Volume 2, pp. 1278-1283. Honolulu, USA.

Yang, M., H. Huang, and G. Xiao (2009). A novel dynamic particle swarm optimization algorithm based on chaotic mutation. *Proceedings of the 2nd International Workshop on Knowledge Discovery and Data Mining (KDD 2009)*. pp. 656-659. Moscow, Russia.

Yao, X., Y. Liu, and G. Lin (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3(2), 82-102.

Yue-Lin, G., A. Xiao-hui, and L. Jun-min (2008). A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. *Proceedings of the Conference on Computational Intelligence and Security (CIS 2008)*. Volume 1, pp. 61-65. Suzhou, China.

Zhan, Z. H., J. Zhang, Y. Li, and H. S. Chung (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39(6), 1362-1381.

Zhan, Z., J. Zhang, and Y. Shi (2010). Experimental study on PSO diversity. *Proceedings of the 3rd International Workshop on Advanced Computational Intelligence (IWACI 2010)*. pp. 310-317. Suzhou, China.
### APPENDIX 1 – COMPARISON TABLES

Table 8: Comparison results GSPO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for 20 particles of 30 dimensions in 3000 iterations

| Test Functions | GSPO | LPSO | DMS-PSO | CLPSO | APSO | DSDPSO |
|----------------|------|------|---------|-------|------|--------|
| Mean           | 1.7916e+01 | 5.7601e-03 | 4.6704e-06 | 2.9291e+00 | 1.4842e-01 | 6.5463e-16 |
| $f_1$          | 7.1447e+01 | 4.1243e-03 | 6.1429e-06 | 9.6072e+01 | 1.7126e-01 | 1.3487e-15 |
| P-value        | 2.7607e-01 | 5.3477e-06 | 3.0037e-03 | 1.8867e-01 | 1.0176e-03 | 4.2828e-02 |
| Mean           | 3.0219e+02 | 1.0972e+02 | 9.9408e+01 | 1.9861e+04 | 3.2302e+02 | 1.5004e-07 |
| $f_2$          | 8.2553e+01 | 2.9632e+00 | 4.1960e+01 | 5.8877e+03 | 8.8822e+01 | 1.3170e-07 |
| P-value        | 1.1709e-12 | 9.5519e-13 | 2.0578e-09 | 4.9760e-12 | 1.3160e-12 | 6.4330e-05 |
| Mean           | 4.3542e+02 | 3.7641e+02 | 2.7344e+02 | 3.0351e+04 | 5.3319e+02 | 1.3198e-06 |
| $f_3$          | 8.3590e+01 | 9.8090e+01 | 1.0264e+02 | 1.7848e+04 | 7.6539e+01 | 1.4009e-06 |
| P-value        | 1.9568e-15 | 5.0453e-13 | 2.9259e-10 | 3.5239e-07 | 8.9328e-18 | 4.7074e-04 |
| Mean           | 5.4786e-02 | 4.7175e-02 | 1.6212e-02 | 3.0641e-01 | 5.9757e-02 | 2.1017e-03 |
| $f_4$          | 1.2444e-02 | 9.7666e-03 | 4.9510e-03 | 2.1417e-01 | 1.2852e-02 | 4.8419e-04 |
| P-value        | 4.2310e-14 | 7.8161e-15 | 8.3921e-12 | 3.8940e-06 | 1.5684e-14 | 9.5707e-10 |
| Mean           | 3.4195e+0405 | 1.2105e+02 | 7.9335e+01 | 1.6060e+06 | 2.5087e+02 | 2.9116e+01 |
| $f_5$          | 7.1928e+04 | 5.8387e+01 | 4.9078e+01 | 3.8732e+06 | 1.9412e+02 | 2.1382e+01 |
| P-value        | 4.6824e-02 | 1.7544e-08 | 7.2942e-07 | 7.9286e-02 | 1.4381e-01 | 7.4264e-06 |
| Mean           | 4.0397e+03 | 3.8723e+03 | 3.3921e+03 | 1.1476e+02 | 1.8532e+03 | 2.7444e+03 |
| $f_6$          | 9.7439e-02 | 3.5625e+02 | 5.9349e+02 | 1.1291e+02 | 1.8849e+03 | 4.2255e+02 |
| P-value        | 1.2563e-13 | 2.1161e-21 | 3.5355e-16 | 2.2105e-04 | 3.0991e-04 | 3.3052e-17 |
| Mean           | 2.2738e+01 | 3.7777e+01 | 1.8905e+01 | 3.0014e+00 | 2.8970e+01 | 2.6645e-16 |
| $f_7$          | 5.2901e+00 | 7.6691e+00 | 5.1155e+00 | 2.8917e+00 | 8.8641e+00 | 6.5076e-16 |
| P-value        | 6.5425e-14 | 5.4632e-15 | 9.8853e-13 | 1.7781e-04 | 8.6769e-12 | 8.2814e-02 |
| Mean           | 2.5125e+01 | 4.7025e+01 | 2.3989e+01 | 5.2439e+00 | 2.8130e+00 | 2.9390e+00 |
| $f_8$          | 6.0741e+00 | 1.2448e+01 | 4.9217e+00 | 5.3572e+00 | 1.0252e+01 | 3.2009e+00 |
| P-value        | 1.3094e-13 | 6.6809e-13 | 7.1008e-15 | 3.2385e-04 | 1.7760e-10 | 6.0109e-04 |
| Mean           | 4.5303e+00 | 2.7555e+00 | 1.9180e+00 | 9.8456e+00 | 4.8676e+00 | 3.3620e-01 |
| $f_9$          | 1.1136e+00 | 1.6614e+00 | 7.7884e-01 | 4.8851e-01 | 4.2714e-01 | 2.7241e-01 |
| P-value        | 1.7682e-13 | 5.0549e-07 | 1.4605e-06 | 1.7908e-26 | 3.3654e-02 | 6.5255e-05 |
| Mean           | 3.0809e-01 | 1.2467e-02 | 1.0320e-02 | 1.8416e-02 | 2.8663e-01 | 1.8158e-01 |
| $f_{10}$       | 3.8386e-01 | 1.7777e-02 | 1.8416e-02 | 2.6653e+00 | 1.8158e-01 | 2.1698e-02 |
| P-value        | 1.9550e-03 | 5.4366e-03 | 2.1462e-02 | 5.1211e-03 | 3.7932e-03 | 2.0376e-04 |
| Mean           | 5.2502e-01 | 8.2130e-03 | 1.4915e-04 | 9.1074e-01 | 1.7303e-02 | 1.1623e-09 |
| $f_{11}$       | 8.4903e-01 | 8.8127e-03 | 1.8565e-04 | 1.0699e+00 | 1.4203e-02 | 9.0222e-10 |
| P-value        | 1.5690e-02 | 5.2227e-04 | 1.9023e-03 | 1.1913e-03 | 2.8975e-05 | 1.4964e-05 |
| Mean           | 4.4746e+01 | 9.0639e+01 | 3.1839e+01 | 6.6216e+02 | 8.3281e+01 | 2.8854e+00 |
| $f_{12}$       | 2.2894e+01 | 2.4041e+01 | 8.0413e+00 | 8.8309e+01 | 3.1386e+01 | 5.8632e+00 |
| P-value        | 3.0212e-12 | 6.9217e-13 | 2.8972e-13 | 2.2654e-18 | 3.1281e-10 | 4.0315e-02 |
Table 9: Comparison results of GPSO, LPSO, DMS-PSO, CLPSO, APSO and DSDPSO for 50 particles of 50 dimensions in 10000 iterations

| Test Functions | GPSO | LPSO | DMS-PSO | CLPSO | APSO | DSDPSO |
|----------------|------|------|---------|-------|------|--------|
| Mean           | 3.6408e+01 | 1.1709e-06 | 8.6047e-13 | 1.6007e-12 | 5.7967e+00 | 6.8476e-23 |
| f1 Std. Dev    | 1.2188e+02 | 9.3600e-07 | 2.2213e-12 | 6.0815e-12 | 2.3336e+01 | 2.4348e-22 |
| P-value        | 1.9663e-01 | 2.1459e-05 | 9.9407e-02 | 2.5371e-01 | 2.8049e-01 | 2.2372e-01 |
| Mean           | 9.2720e+02 | 3.3203e+02 | 3.4948e+02 | 3.0310e+04 | 1.0761e+03 | 1.9793e-09 |
| f2 Std. Dev    | 1.1121e-02 | 9.7868e+01 | 1.1019e+02 | 1.6111e+04 | 1.5859e+02 | 7.8385e-09 |
| P-value        | 3.1056e-19 | 4.5024e-12 | 1.4669e-11 | 7.8642e-08 | 1.4614e-17 | 2.7284e-01 |
| Mean           | 1.2430e+03 | 1.4379e+03 | 7.4146e+02 | 6.2315e+04 | 1.4860e+03 | 3.6115e+11 |
| f3 Std. Dev    | 1.4253e+02 | 1.5402e+02 | 1.9543e+02 | 8.5152e+04 | 2.2121e+02 | 8.3015e-11 |
| P-value        | 1.3360e-19 | 3.7150e-20 | 5.7934e-13 | 4.0035e-03 | 1.7631e-17 | 6.6651e-02 |
| Mean           | 1.4051e-01 | 8.9698e-02 | 2.9690e-02 | 3.2467e-01 | 1.8012e-01 | 2.7437e-04 |
| f4 Std. Dev    | 2.6527e-02 | 1.8103e-02 | 6.6752e-03 | 2.1208e-01 | 2.1536e-02 | 1.3640e-04 |
| P-value        | 1.4387e-15 | 4.9015e-16 | 4.5136e-14 | 1.5604e-06 | 2.9286e-19 | 2.8142e-08 |
| Mean           | 4.0587e+05 | 1.0355e+02 | 8.9578e+01 | 2.8716e+02 | 3.8231e+04 | 8.2594e+00 |
| f5 Std. Dev    | 2.6389e+05 | 3.8853e+01 | 5.1377e+01 | 4.4957e+02 | 1.4680e+05 | 4.9569e+00 |
| P-value        | 1.4632e-06 | 2.9064e-10 | 2.4485e-07 | 1.2011e-02 | 2.5857e-01 | 4.7331e-07 |
| Mean           | 8.1179e+03 | 6.9191e+03 | 5.9235e+03 | 3.6383e-04 | 7.3343e+03 | 4.7493e+03 |
| f6 Std. Dev    | 2.3530e+03 | 4.5626e+02 | 7.5986e+02 | 6.9101e-12 | 6.0505e+03 | 4.9467e+02 |
| P-value        | 3.3496e-12 | 3.9191e-24 | 1.0947e-18 | 5.3075e-153 | 3.1316e-05 | 2.1929e-20 |
| Mean           | 2.8405e+01 | 5.9188e+01 | 1.8755e+01 | 4.9504e-12 | 3.1801e+01 | 0.0000e+00 |
| f7 Std. Dev    | 5.9531e+00 | 1.1344e+01 | 3.7126e+00 | 7.2157e-12 | 1.3376e+01 | 0.0000e+00 |
| P-value        | 9.7783e-15 | 1.8966e-15 | 3.4383e-15 | 6.3285e-03 | 1.9439e-09 | NaN |
| Mean           | 2.9597e+01 | 6.9303e+01 | 2.3510e+01 | 4.3004e-08 | 3.5759e+01 | 8.8818e-17 |
| f8 Std. Dev    | 5.5392e+00 | 1.1018e+01 | 6.7069e+00 | 5.2497e-08 | 1.1294e+01 | 3.9721e+16 |
| P-value        | 1.2000e-15 | 5.9973e-17 | 2.5280e-12 | 1.6516e-03 | 1.5100e-11 | 3.2988e-01 |
| Mean           | 2.8477e+00 | 1.6635e+00 | 1.2932e+00 | 1.2169e+01 | 4.7056e+01 | 1.4856e-01 |
| f9 Std. Dev    | 1.5365e+00 | 1.0521e-01 | 1.1422e+01 | 3.4260e-01 | 7.3243e-01 | 2.5570e-01 |
| P-value        | 9.8635e-08 | 1.7782e-24 | 9.7970e-22 | 1.6480e-30 | 4.0477e-17 | 1.8178e-02 |
| Mean           | 5.9085e-01 | 4.9700e-04 | 1.0078e-02 | 2.7391e-02 | 5.6336e-02 | 1.9227e-02 |
| f10 Std. Dev   | 9.9017e-01 | 2.2119e-03 | 1.1828e-02 | 2.1163e-02 | 8.1117e-02 | 2.5049e-02 |
| P-value        | 1.5182e-02 | 3.2928e-01 | 1.1816e-03 | 1.4118e-05 | 5.8818e-03 | 2.7904e-03 |
| Mean           | 1.2494e+00 | 6.2859e-05 | 1.0619e-07 | 8.9149e-09 | 1.2521e-01 | 1.2434e-13 |
| f11 Std. Dev   | 1.8753e+00 | 1.8773e-05 | 1.6659e-07 | 8.4447e-09 | 4.5512e-01 | 8.9774e-14 |
| P-value        | 7.7015e-03 | 5.6734e-12 | 1.0230e-02 | 1.4866e-04 | 2.3360e-01 | 5.9585e-06 |
| Mean           | 6.0300e+01 | 1.2462e+02 | 2.6565e+01 | 1.1046e+03 | 8.2734e+01 | 5.3291e-16 |
| f12 Std. Dev   | 1.6712e+01 | 1.6415e+01 | 6.0228e+00 | 1.0434e+02 | 2.3265e+01 | 8.3518e-16 |
| P-value        | 1.5131e-12 | 1.7970e-18 | 4.0924e-14 | 3.4779e-21 | 1.9582e-12 | 1.0163e-02 |