

A New Particle Swarm Optimization Method Enhanced With a Periodic Mutation Strategy and Neural Networks

Y. Volkan Pehlivanoglu

Abstract—Particle swarm optimization (PSO), a relatively new population-based intelligence algorithm, exhibits good performance on optimization problems. However, during the optimization process, the particles become more and more similar, and gather into the neighborhood of the best particle in the swarm, which makes the swarm prematurely converged most likely around the local solution. A new optimization algorithm called multifrequency vibrational PSO is significantly improved and tested for two different test cases: optimization of six different benchmark test functions and direct shape optimization of an airfoil in transonic flow. The algorithm emphasizes a new mutation application strategy and diversity variety, such as global random diversity and local controlled diversity. The results offer insight into how the mutation operator affects the nature of the diversity and objective function value. The local controlled diversity is based on an artificial neural network. As far as both the demonstration cases' problems are considered, remarkable reductions in the computational times have been accomplished.

Index Terms—Airfoil, diversity, mutation, neural nets, particle swarm optimization (PSO).

I. INTRODUCTION

AS IN OTHER evolutionary algorithms (EAs), the particle swarm optimization (PSO) method is a population-based stochastic optimization algorithm that originates from nature. These algorithms often require more cost function evaluations than comparable gradient-based algorithms. However, they provide attractive features such as ease of implementation for both continuous and discrete problems, efficient use of large numbers of parallel processors, no requirement for continuity in response functions, and more robust solution generations for searching global or near global solutions. PSO algorithms search the optimum within a population called a swarm. PSO benefits from two types of learning: cognitive learning based on an individual's history and social learning based on a swarm's history accumulated by sharing information among all particles in the swarm. Since its development in 1995 by Kennedy *et al.* [1], it has attracted significant attention. Most of the investigations on this topic are related to either the mathematical analysis focusing on how it works, or improving

the PSO to get faster and more reliable solutions. The impetus for the latter is typically the trapping of the solution candidates to local optima, or the so-called premature convergence.

An optimization algorithm prematurely converges to a local optimum if it is no longer able to explore sections of the search space other than the examined area, and another region exists that contains a solution superior to the currently discovered one. The main reason for the premature convergence might be the lack of diversity. Diversity is essential for a healthy search; and mutations are the basic operators to provide the necessary variety within a swarm. Apart from designing new mutation operators, researchers have put relatively less effort into investigating how to apply mutation operators during the process and determining what kind of diversity should be provided within the swarm in the PSO process. Therefore, after having close scrutiny of the diversity concept based on quantification and qualification studies, this paper introduces new and improved mutation strategies and operators to provide beneficial diversity within the swarm. This new approach is called multifrequency vibrational PSO. The new approach has been applied to selected benchmark test functions and direct shape optimization of an airfoil in transonic flow. As demonstrated by these test cases, it was observed that the new algorithms outperform the current popular algorithms. The principal role of this new approach was to determine which particles should be mutated, when they should be mutated, and which ones should be merged into the swarm. The new mutation operators, when combined with a mutation strategy and an artificial intelligent method such as neural networks, provided local and global diversities. The new approach also introduced random and controlled diversity.

II. DEFINITION AND MATHEMATICAL ANALYSIS OF BASIC PSO ALGORITHM

Let s be the swarm size, d be the particle dimension space, and each particle of the swarm have a current position vector X_i , a current velocity vector V_i , and an individual best position vector P_i found by the particle itself. The swarm also has the global best position vector P_g found by any particle during all prior iterations in the search space. Assuming that the function f is to be minimized, and describing the following notations in t th iteration, the definitions are as follows:

$$\begin{aligned} X_i(t) &= (x_{i,1}(t), x_{i,2}(t), \dots, x_{i,d}(t)) \\ x_{i,j}(t) &\in R^d, i = 1, 2, \dots, s \end{aligned} \quad (1)$$

Manuscript received March 22, 2010; revised January 21, 2011 and December 10, 2011; accepted April 14, 2012. Date of publication June 1, 2012; date of current version May 24, 2013.

The author is with the Department of Aerospace Engineering, Turkish Air Force Academy, Yesilyurt, Istanbul 34149, Turkey (e-mail: vpehlivan@hho.edu.tr).

Digital Object Identifier 10.1109/TEVC.2012.2196047

where each dimension of a particle is updated using the following equations:

$$\begin{aligned} v_{i,j}(t) &= v_{i,j}(t-1) \\ &+ c_1 r_1 (p_{i,j}(t-1) - x_{i,j}(t-1)) \\ &+ c_2 r_2 (p_{g,j}(t-1) - x_{i,j}(t-1)) \end{aligned} \quad (2)$$

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t). \quad (3)$$

In (2), c_1 and c_2 denote constant coefficients and r_1 and r_2 are elements from random sequences in the range of (0, 1). The parameter c_1 controls the influence degree of a cognitive part of an individual, and c_2 determines the effect of a social part of the swarm. The personal best position vector of each particle is computed using the following expression:

$$\mathbf{P}_i(t) = \begin{cases} \mathbf{P}_i(t-1) & \text{if } f(\mathbf{X}_i(t)) \geq f(\mathbf{P}_i(t-1)) \\ \mathbf{X}_i(t) & \text{if } f(\mathbf{X}_i(t)) < f(\mathbf{P}_i(t-1)) \end{cases}. \quad (4)$$

Then, the global best position vector is found by

$$\mathbf{P}_g(t) = \arg \min_{\mathbf{P}_i(t)} f(\mathbf{P}_i(t)). \quad (5)$$

Ozcan and Mohan [2] analyzed updating (2) and (3) in an analytical way. Let us reissue the analytical analysis to go a step further. For the sake of simple mathematical analysis, assume that c_1 , r_1 and c_2 , r_2 are constants that are equal to φ_1 and φ_2 , respectively, and $\mathbf{P}_i(t)$, $\mathbf{P}_g(t)$ are also constants, such as p_i , and p_g , respectively. Then, (2) and (3) become

$$\begin{aligned} v_{i,j}(t) &= v_{i,j}(t-1) \\ &+ \varphi_1 (p_i - x_{i,j}(t-1)) + \varphi_2 (p_g - x_{i,j}(t-1)) \end{aligned} \quad (6)$$

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t) \quad (7)$$

by getting $x_{i,j}(t-1)$ using (7), and substituting it into (6) we get the following particle path equation:

$$\begin{aligned} x_{i,j}(t) - (2 - \varphi_1 - \varphi_2)x_{i,j}(t-1) \\ + x_{i,j}(t-2) = \varphi_1 p_i + \varphi_2 p_g \end{aligned} \quad (8)$$

$$x - \alpha \dot{x} + \ddot{x} = \beta \quad (9)$$

$$\alpha = (2 - \varphi_1 - \varphi_2) \quad (10)$$

$$\beta = \varphi_1 p_i + \varphi_2 p_g. \quad (11)$$

This is a linear nonhomogeneous second-order differential equation. The general solution of (8) can be derived by getting complementary and particular solutions. The resulting solution, including initial boundary conditions, can be found such that

$$x(t) - \chi \tau^t + \varepsilon \phi^t + \gamma \quad (12)$$

where

$$\chi = \left(0.5 - \frac{(\varphi_1 + \varphi_2)}{2\eta} \right) x_{i,j}^{(0)} + \frac{v_{i,j}^{(0)}}{\eta} + \frac{\beta}{2\eta} - \frac{\beta}{2(\varphi_1 + \varphi_2)} \quad (13)$$

$$\varepsilon = \left(0.5 + \frac{(\varphi_1 + \varphi_2)}{2\eta} \right) x_{i,j}^{(0)} - \frac{v_{i,j}^{(0)}}{\eta} - \frac{\beta}{2\eta} + \frac{\beta}{2(\varphi_1 + \varphi_2)} \quad (14)$$

$$\tau = \frac{\alpha + \eta}{2} \quad (15)$$

$$\phi = \frac{\alpha - \eta}{2} \quad (16)$$

$$\eta = \sqrt{\alpha^2 - 4} \quad (17)$$

$$\gamma = \frac{\beta}{\varphi_1 + \varphi_2}. \quad (18)$$

This result may expose several different cases depending on the α value, specifically the sum of φ_1 and φ_2 values. The value of η becomes a complex number for $0 < \varphi_1 + \varphi_2 < 4$. Converting ϕ and τ into polar forms and substituting them into (12), we can obtain the following simplified equations:

$$x_{i,j}(t) = \kappa \sin(\theta t) + \lambda \cos(\theta t) + \gamma \quad (19)$$

$$\theta = \tan^{-1} \left(\frac{||\eta||}{|\alpha|} \right) \quad (20)$$

$$\kappa = \frac{2v_{i,j}^{(0)} - (\varphi_1 + \varphi_2)x_{i,j}^{(0)} + \beta}{||\eta||} \quad (21)$$

$$\lambda = x_{i,j}^{(0)} - \gamma. \quad (22)$$

This is the general form for the particle path. We can make some assumptions such that there is only one dimension, $p = p_i = p_g$, $\varphi_1 + \varphi_2 = \varphi$ and $2 > \varphi > 0$ or $4 > \varphi > 2$, $x^{(0)} = x_0$, $v^{(0)} = v_0$. Then, the path of any particle in the swarm is directed by the following equations:

$$x(t) = A_1 \sin(\theta t) + A_2 \cos(\theta t) \quad (23)$$

$$A_1 = \frac{2v_0 - \varphi x_0 - \varphi p}{\sqrt{|\varphi^2 - 4\varphi|}} \quad (24)$$

$$A_2 = x_0 - p \quad (25)$$

$$\frac{\sqrt{|\varphi^2 - 4\varphi|}}{|2 - \varphi|}. \quad (26)$$

Clerc and Kennedy [3] concluded that the particle, as seen in discrete time “surfs” on an underlying continuous foundation of sine waves, jumps from one wave to another by using random numbers in the periphery of a random number distribution. At this point, let us proceed with (23) and add the redefined Kronecker delta function as follows:

$$x(t) = \begin{cases} x(t) + A_n \delta & \delta = 1 \text{ if } t \geq nf \quad n = 1, 2, \dots \\ x(t) & \delta = 0 \text{ if } t < nf \end{cases}. \quad (27)$$

Here, f is the impulse period or frequency and A_n is the third amplitude that is either negative or positive. The resulting graphs for (23) and (27) with positive A_n and f with 50 can be plotted in Fig. 1. The purpose of adding an impulse depending on the time frame is to catch the big wave, or in other words, to extend the search domain from one local area to another local area for the particle. The effect of an impulse will be analyzed in subsequent sections.

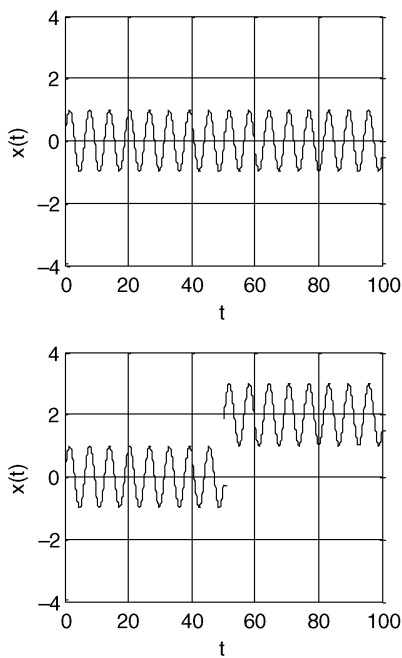


Fig. 1. Trajectory of a particle in a simplified PSO algorithm with (23) (top) and (27) (bottom).

III. IMPROVED PSO ALGORITHMS IN THE LITERATURE

PSO may have some issues related to convergence speed, prematurely converged solutions, deficient accuracy, or lack of diversity. Numerous modifications have been proposed so far to overcome these issues. Valle *et al.* [4] gave a detailed overview of the basic concepts of PSO and its variations. Improvement methods may be broadly divided into two categories focusing on either hardware or software improvements. The hardware focus is related to parallel computing. The software studies may be further classified as hybridization with other search algorithms, inspiration from other stochastic-based algorithms, rearranging and manipulating the updating equation descriptions mainly related to the velocity equation, parameter and neighborhood topology manipulations within the updating phase, and finally the swarm sizing and grouping.

Since each particle's cost function can be evaluated independently in the swarm, PSO is ideally suited for synchronized or unsynchronized execution on a cluster of computers in parallel. Integrating PSO algorithm with other search algorithms is called hybridization. A common hybrid application is to use PSO with another population-based algorithm such as a genetic algorithm (GA). Yet another technique is to use a gradient-based algorithm as an integrated part. Manipulating or rearranging (2) and (3) is an important part of algorithm improvements. In these manipulations, the velocity equation is extended with an extra term or shortened depending on the approach. Parameter redesigns constitute another bundle of modification packages. Since Shi *et al.* [5] suggested the use of an inertia weight (w) multiplying by the previous velocity in (2), the parameter number to be adjusted is increased to three. The inertia weight determines the effect of a previous velocity. The common PSO parameters, w , c_1 , c_2 may be constant, linearly or nonlinearly changeable, and let us say periodically, adaptively, chaotic, or randomly changeable depending on the

time or other reference such as cost function value, velocity, or metropolis criteria. Neighborhood topology originates from the selection of position vectors P_i and P_g . In the original algorithm, the particle is going to be attracted by the individual best position and global best position. This means that no other individual best position vectors will have an attractive effect on the individual itself, and also the best one will never be changed at that iteration. At this point, different neighborhood topology descriptions are introduced. Population itself is also studied in many different aspects. Dynamic swarm sizing or dividing the swarm into subgroups may be promising to solve some of the optimization problems. Inspirations from other stochastic-based algorithms, such as GA, simulated annealing, or opposition-based learning algorithms, are other popular methodologies to improve the basic PSO algorithm. Commonly used GA reproduction operators, including selection, crossover, elitism, and mutation, can be deployed in the PSO algorithm architecture.

Among GA operators, mutation is the most utilized, because the drawback of PSO is due to the lack of diversity, which forces swarm particles to converge to the position found so far that may be a local optimum. Without an effective diversity enhancing mechanism, the optimization algorithm may not be able to efficiently explore and exploit the search space. Mutation operators introduce new particles into a swarm by manipulating a current particle, thus adding diversity to the swarm and likely preventing stagnation of the search in local optima. However, the mutation application procedure brings some new adjustments to the algorithm, such as the criteria of mutation applications, the position where mutation will be applied, and the selection of random probability distribution sequence. Mutation probability percentages, similarity, and closeness to each other in terms of Euclidian distance, and describing diversity thresholds may be the criteria for mutation applications. Gaussian, Cauchy, Beta distributions, chaotic distribution based on logistic maps, or mixed types such as cloud distribution are possible random distribution sequences. Randomly selected current positions, velocities, or even individual best position and global best position vectors may be mutated [6]–[17]. Higashi and Iba [6] gave an early usage of mutation operator that changes a particle dimension value using a random number drawn from a Gaussian distribution. Stacey *et al.* [7] applied a similar mutation operator with a random number based on a Cauchy probability distribution to a particle to be mutated. Instead of using the previous random probability distributions, Pant *et al.* [8] defined a Sobol mutation operator that uses quasirandom sequences that cover the search domain more evenly in comparison to the random probability distributions. Zavala *et al.* [9] implemented two different perturbation operators to keep diversity in a swarm. In their study, C-perturbation and M-perturbation operators were applied to an individual best position vector instead of perturbing the position of the particle. Jia *et al.* [10] proposed a chaotic and Gaussian mutation PSO to solve the premature and low optimizing precision in standard PSO. In the earlier iterative phase of the optimization process, chaotic mutation was introduced for the global search, and then Gaussian mutation was incorporated into PSO to improve the local exploitation

quality. Wu *et al.* [11] preferred use of a novel mutation operator called cloud mutation in terms of the properties of randomness and stable tendency of normal cloud model. Chen *et al.* [12] restricted the possible particle to be mutated to the global best position vector found by any particle during all prior iterations in the search space. They introduced a mutation operator with adaptive mutation probability into the algorithm. Example attempts for PSO using chaos methods were given by [13] and [14]. Liu *et al.* [13] incorporated chaos into PSO with an adaptive inertia weight factor to construct a chaotic PSO. Yang *et al.* [14] combined the chaotic mutation operator with the linear decreasing inertia weight. The chaotic mutation operator was introduced to enhance the performance of the local search ability and to improve the search precision of the algorithm. In both studies, a logistic map is adopted as a chaos model in mutation operators. Liu *et al.* [15] proposed an improved PSO with mutation based on collectivity. A collectivity is a concept based on similarity between a particle and a current global best particle in the swarm and was used to mutate the position of the particles. To avoid unnecessary iterations in every local optimum, Biao *et al.* [16] developed mutated fast position convergence PSO. In this approach, a mutation operation is applied only if it is necessary. Luitel and Venayagamoorthy [17] improved a hybrid particle swarm optimization with the quantum infusion algorithm. In this approach, a randomly chosen particle from the individual best position vectors is utilized to carry out the quantum operation, and thus generates an offspring by mutating the global best position vector.

Although mutations provide diversity within the swarm, they may also cause a peripheral stagnation search. Therefore, the type of mutation operator and its application strategy are important decision parts for the mutation applications. Apart from designing new mutation operators, researchers have put relatively less effort into investigating how to apply mutation operators during the process and determining what kind of diversity should be provided within the swarm in the PSO process.

A. Improved PSO Algorithms

When a new algorithm is proposed in the literature, it is expected to be compared with the current state-of-the-art algorithms on commonly used benchmark test functions. Obviously, it is difficult to make a fair comparison, because each algorithm may have peculiar tuning, which results in important differences on the cost functions. However, well-defined and straightforward algorithms may provide a relatively good inference. Four well-known PSO algorithms are selected as comparative optimization algorithms. These are constriction factor PSO (c-PSO), linearly decreased inertial weight PSO (w-PSO), Gaussian mutation-based PSO (g-PSO), and comprehensive learning PSO (cl-PSO).

1) *c-PSO*: The particle swarm with a constriction factor was introduced by Clerc and Kennedy [3], which investigated the use of a parameter called the constriction factor. With the constriction factor K , the particle velocity and position dimensions are updated via

$$\begin{aligned} v_{i,j}(t) = & K((v_{i,j}(t-1) + c_1 r_1(p_{l,j}(t-1) - \\ & X_{i,j}(t-1)) + c_2 r_2(p_{g,j}(t-1) - X_{i,j}(t-1))) \end{aligned} \quad (28)$$

$$\begin{aligned} K = & \frac{2}{|2 - \psi - \sqrt{\psi^2 - 4\psi}|} \\ \psi = & c_1 + c_2, \psi > 4 \\ x_{i,j}(t) = & x_{i,j}(t-1) + v_{i,j}(t). \end{aligned} \quad (29)$$

A particularly important contribution of this factor is that if it is correctly chosen, it guarantees the stability of PSO without the need to bind the velocities. Typically, values of 2.05 are used for c_1 and c_2 , making ψ equal to 4.1 and K equal to 0.729.

2) *w-PSO*: Shi and Eberhart [18] introduced the idea of a time-varying inertia weight. The idea was based on the control of the diversification and intensification behavior of the algorithm. The velocity is updated in accordance with the following expressions:

$$\begin{aligned} v_{i,j}(t) = & w(t)v_{i,j}(t-1) \\ & + c_1 r_1(p_{l,j}(t-1) - x_{i,j}(t-1)) \\ & + c_2 r_2(p_{g,j}(t-1) - x_{i,j}(t-1)) \\ x_{i,j}(t) = & x_{i,j}(t-1) + v_{i,j}(t). \end{aligned} \quad (30)$$

The inertia weight w is decreased linearly starting from the initial point w_{ini} and ending to the last point w_{end} related to the maximum iteration number T . Normally, the starting value of the inertia weight is set to 0.9 and the final to 0.4. However, we tuned them to the [0.6, 0.2] range for better performance.

3) *g-PSO*: First Gaussian mutation based PSO algorithm is introduced by Higashi and Iba [6], which used a mutation operator that changed a particle dimension value using a random number drawn from a Gaussian distribution. A particle is selected for mutation using a mutation rate that is linearly decreased during a run. However, it is improved by several researchers. Pant *et al.* [19] developed the Gaussian mutation operator application technique for updating the position of the swarm particles. The mutation operator is activated only when the diversity of the swarm becomes less than a certain threshold d_{low} . The velocity is updated via (30) in addition to the following criteria:

$$\text{if } d < d_{\text{low}} \text{ then } x_{i,j}(t) = x_{i,j}(t-1) + \xi \text{ rand} \quad (31)$$

$$d = \frac{1}{s} \sum_{i=1}^s \sqrt{\sum_{j=1}^d (x_{i,j}(t-1) - \bar{x}_j(t-1))^2} \quad (32)$$

$$\bar{x}_j(t-1) = \sum_{i=1}^s \frac{x_{i,j}(t-1)}{s} \quad (33)$$

where ξ is the scaling factor and *rand* is a random number generated by Gaussian distribution. The diversity description given by (32) is based on the differences between the average dimensional position and the current dimensional particle positions.

4) *cl-PSO*: The comprehensive learning particle swarm optimizer was proposed by Liang *et al.* [20], which used all other particles' historical best information to update a particle's velocity. This approach keeps the diversity of the swarm in a high level to be preserved to discourage the

premature convergence. A particle's velocity and its position are updated by the following equations:

$$\begin{aligned} v_{i,j}(t) &= w(t)v_{i,j}(t-1) + cr(p_{I(f_i(j))},j(t-1) - x_{i,j}(t-1)) \\ x_{i,j}(t) &= x_{i,j}(t-1) + v_{i,j}(t) \end{aligned} \quad (34)$$

where f_i defines which particles' best position vector the particle i should follow, c is the constant value, and r is a random number drawn from a random sequence in the range of $(0, 1)$. The decision about f_i depends on the learning probability value, Pc_i which is defined as follows:

$$Pc_i = 0.05 + 0.45 \frac{\left(\exp\left(\frac{10(i-1)}{s-1}\right) - 1 \right)}{(\exp(10) - 1)}. \quad (35)$$

For each dimension of particle i , a random number is generated and compared with the value of Pc_i . If a random number is larger than the learning probability value, the related dimension will learn from its own best position vector; otherwise, it will learn from another particle's P_i . A tournament selection procedure is taken into consideration to determine the particle i . The inertia weight w is decreased linearly starting from the initial point w_{ini} , and ending to the last point w_{end} , related to the maximum iteration number T .

B. Quantification in a Simple Test Case

The word *diversity* means "the condition of being diverse, variety," or "the quality or state of being composed of many different elements or types." Mathematically, diversity refers to Euclidian distance among the members in a group. It is possible to describe different diversity metrics related to a design variable, a particle, and a swarm. Here, the diversity description given by (32) and (33) is used.

It may be beneficial to see the change of swarm diversity for a better understanding of algorithm characters. For this reason, a simple test case is constructed. One of the benchmark test functions, the Ackley test function, is selected as a test case and selected PSO algorithms are compared in terms of generation and swarm diversity values. The Ackley test function is a multimodal test function and is described in Table I.

In order to reduce the computational cost, the problem dimension d is selected as 3; the swarm size s is selected as 5; and the maximum generation T is fixed to 200. The experimental setups for c-PSO, w-PSO, g-PSO, and cl-PSO are given in Table I. All algorithms are run 40 times and the results are averaged. The resulting plots can be seen in Fig. 2. We can see the averaged best (elite) f_{min} values versus generations on the upper side and the swarm diversity values ($D_{S,t}$) versus generations on the lower side.

Fig. 2 shows that g-PSO outperforms the others. At the end of the maximum generation in g-PSO, the objective function reaches a value close to 10^{-4} . However, the other three algorithms are about the level of 10^{-1} for objective function values. It seems that the threshold diversity-based mutation applications decreased by 40% the required generation number.

TABLE I
FEATURES OF PSO ALGORITHMS

Algorithm	w	c	c_1	c_2	d_{low}	ξ
c-PSO	—	—	2.05	2.05	—	—
w-PSO	0.6–0.2	—	2	2	—	—
g-PSO	0.6–0.2	—	2	2	10^{-3}	6
cl-PSO	0.9–0.4	1.49445	—	—	—	—

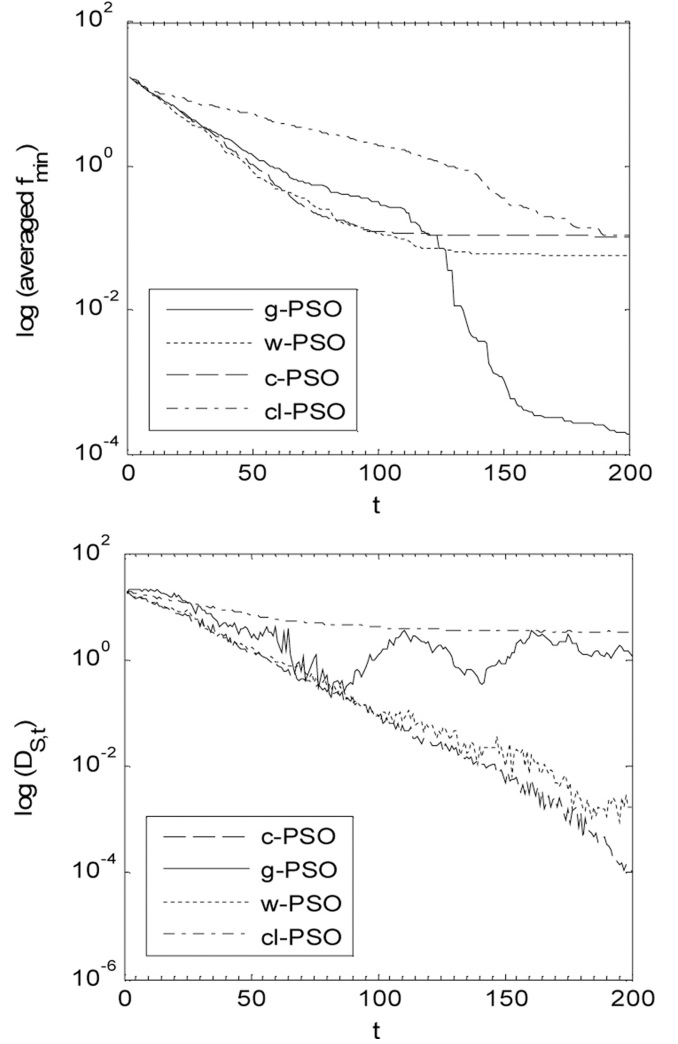


Fig. 2. Comparative results of selected PSO algorithms.

On the other hand, the swarm diversity values have different characteristics. Both constriction factor based PSO and linearly decreased inertia weight based PSO have almost linearly decreased swarm diversities in the logarithmic scale. As the generations of particles in the swarm evolve, they lose their diversity and become similar to each other. Comprehensive learning PSO shows an expected behavior. It keeps the level of swarm diversity in high degrees. However, g-PSO displays a different characteristic that distinguishes it from the other algorithms. The algorithm decreases the swarm diversity until a certain generation number. When the mutation operator is activated, the swarm diversity is increased and the diversity level is kept on a threshold diversity level with a certain periphery. Another interesting point inferred from Fig. 2 is

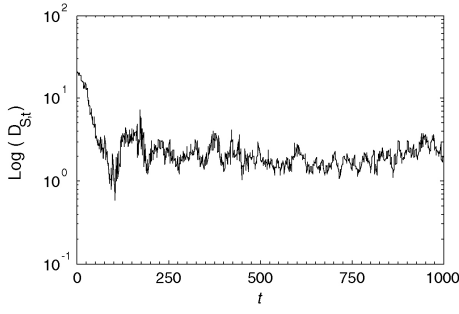


Fig. 3. Swarm diversity signal in the g-PSO optimization process.

the periodicity. It seems that the determination of the diversity level brings unclear, but dominant periodic mutation applications. The updating equations consume the diversity of the swarm with a certain rate. This observation is more understandable in Fig. 3. However, we need to take a closer look at the periodicity of diversity.

1) *Wavelet Analysis*: The significance of mutation applications that result in unclear periodicity can be searched in more detail by using the wavelet analysis. The swarm diversity, $D_{S,t}(t)$, is a sequence of data points measured at uniform generation intervals, and it can be described as a time series. A wavelet analysis can be applied to the swarm diversity to determine the dominant periods of variability. We can express any time series s' in terms of signals, such as

$$s' = \sum_{j=1}^J D_j \quad (36)$$

$$D_j = \sum_{k=1}^K \alpha_{j,k} \psi_{j,k} \quad (37)$$

where $\alpha_{j,k}$ is a coefficient and $\psi_{j,k}$ is a wavelet function. This expression is usually divided into approximate and detail signals such as

$$s' = A_J + \sum_{j \leq J} D_j \quad (38)$$

where A_J is the approximate signal and D_j are the detail signals [21]. The biggest part of the signal energy is carried by an approximate signal. The remaining energy is shared by detailed signals, sometimes called noise signals. There are different wavelet functions in the literature. Here, Meyer wavelets [22] were used. The wavelet analysis is based on the same optimization problem with $T = 1000$. The whole diversity signal can be seen in Fig. 3.

The diversity change in terms of the last 800 generations is included for a steady result. This signal can be analyzed by using discrete approximation of Meyer wavelets at 9-level. So, the diversity signal is expressed by

$$D_{S,t} = A_9 + \sum_{j=1}^9 D_j. \quad (39)$$

The resulting discrete wavelet transform for the diversity can be seen in Fig. 4. The figure includes only the approximation signal A_9 and the detail signals: D_j , $j = 6, 7, 8$, and 9. The remaining signals are ignored due to noisy natures. The picture

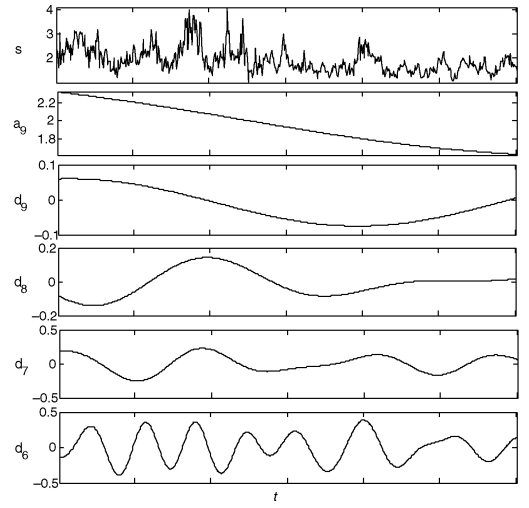


Fig. 4. Wavelet analysis of a part of the swarm diversity signal.

shows that the main diversity signal carries some periodic signals and noises.

The wavelet analysis showed that the description of threshold diversity brings new parameters, such as periodicity and bandwidth, which is a kind of amplitude. Mutation activation that depends on the diversity level may be beneficial since it provides guaranteed diversification. However, finding a proper threshold value may not be easy and may require an extra study. Additionally, a fixed diversity level may not be proper for a more accurate convergence in different optimization problems. Instead of using threshold diversity, we may directly construct the periodicity and amplitude features in terms of a new mutation strategy that includes both of them.

2) *Periodic Mutation Activation and v-PSO*: A generalized mutation operation including a mutation strategy can be described as follows:

$$x_{i,j}(t) = F(g(x_{i,j}(t)), fr) \quad (40)$$

where F is the generalized mutation function, $g(x_{i,j}(t))$ is the mutation operator providing the offspring vector, and fr is a user-defined application frequency. The mutation strategy focuses on investigating how to apply mutation operators during the optimization process. Right after updating applications, in every fr^{-1} period of the generations applying the mutation operator to all particle dimensions of the whole swarm, particles in the swarm spread throughout the design space. This operator is called the global mutation operator and is given by

$$\begin{aligned} x_{i,j}(t) &= x_{i,j}(t)[1 + A(0.5 - rand)\delta] \\ i &= 1, 2, \dots, s \\ j &= 1, 2, \dots, d \\ \delta &= \begin{cases} 1 & \text{if } t = nfr \quad n = 1, 2, \dots \\ 0 & \text{if } t \neq nfr \end{cases} \end{aligned} \quad (41)$$

where A is an amplitude factor defined by the user, and it may be selected as a fixed number or computed during the generations; $rand$ is a real random number specified by a random number generator in accordance with $N[0, 1]$. In the applications, the Gaussian probability density function is used.

TABLE II
FEATURES OF v-PSO ALGORITHM

Algorithm	w	c_1	c_2	fr	A
v-PSO	0.05	2	2	25	1

However, other density functions can also be used in (41). It should be noted that this expression resembles (27). The aim of designing such a mutation strategy is to catch the big wave for escaping from local traps and getting the correct search pattern. The velocity and the positions are updated via (30), except the generations corresponding to the mutation period. This new algorithm is named vibrational PSO (v-PSO).

3) *Comparison of v-PSO in a Simple Case:* The new algorithm is compared with the previous algorithms in the same test function case. In addition to Table I, the features of v-PSO are given in Table II. The results are depicted in Fig. 5. On the upper side of the figure, the averaged best objective function values versus generations are shown. Vibrational PSO outperforms the other algorithms. It decreases the required generations 50% or more as compared with c-PSO, cl-PSO, and w-PSO, and 25% as compared with g-PSO. Additionally, the accuracy of the solution is about 10^{-7} level, and it is almost twice as accurate as g-PSO. On the lower side of the figure, the swarm diversity versus generations is observed. The swarm diversity of v-PSO is gradually decreased during the generations. However, its change is in stages and periodic due to periodic mutation applications.

Fig. 5 also shows that the performance of the algorithm is directly related to the diversity. However, this relation is not only subject to quantification, but also subject to qualification. Therefore, we need to analyze the qualification of the diversity.

Additionally, the particles' trajectories of c-PSO and v-PSO in the 4-D domain can be seen in Fig. 6. In these plots, the x -axis of a particle is ridden on the generations with a certain range, δX . Forty runs in total including 40 000 particle positions are plotted in each figure. The particles that belong to the v-PSO process capture the global optimum during the generations. However, the particles that belong to the c-PSO process look more distracted.

IV. QUALIFICATION OF DIVERSITY

Up to now, the quantification of diversity within the swarm with different swarm algorithms, including a new mutation strategy based PSO was analyzed. It seems that diversification is useful and required for fast convergence. However, high level diversity may sometimes have destructive effects on the swarm [23]. Therefore, we need to analyze the quality of diversity. It is assumed that the quality of diversity can be categorized depending on the distribution character in the search space and the determination process. First, we will focus on the distribution character of the diversity such as global or local diversity, and then the determination process such as random or controlled diversity.

4) *Global and Local Diversity:* If the mutation applications cause a wider range of distributions of the current swarm in the search space, it can be called global diversity. Suppose that the

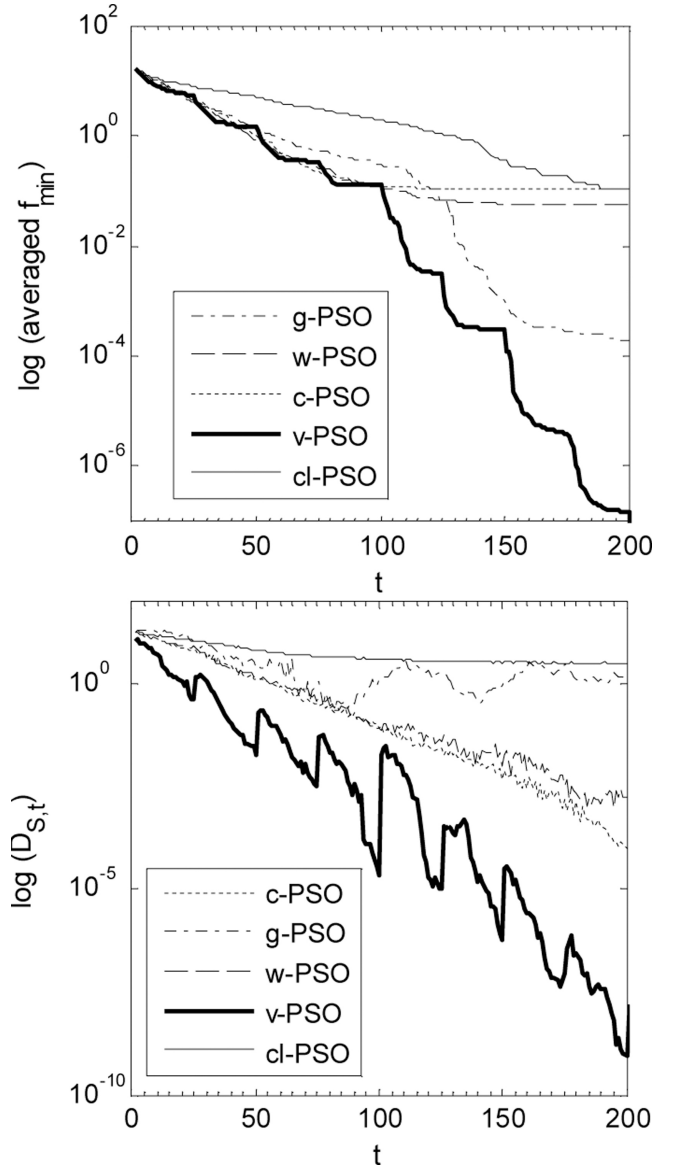


Fig. 5. Comparative results of PSO algorithms.

diversity of the current swarm is described as $D_S(t)$, and the new diversity after mutation applications is called $D_S(t+1)$. If (42) is valid; it means that the mutation applications provide a global diversity in the swarm

$$D_S(t+1) \gg D_S(t). \quad (42)$$

Local diversity concept can be based on a particle and its neighborhood. Assume that we have two particles labeled x_m and x_k in the swarm. The diversities for these particles are named $D_p^m(t)$ and $D_p^k(t)$. Let us describe the local difference in diversity as ΔD_L . If the following criterion:

$$D_p^m(t) - D_p^k(t) < \Delta D_L \quad (43)$$

$$D_{p,t}^{m,k} \rightarrow \Delta D_L \quad (44)$$

is valid for the particles, we can say that there is a local diversity within the swarm. During the optimization process, local diversity is going to be a common phenomenon within

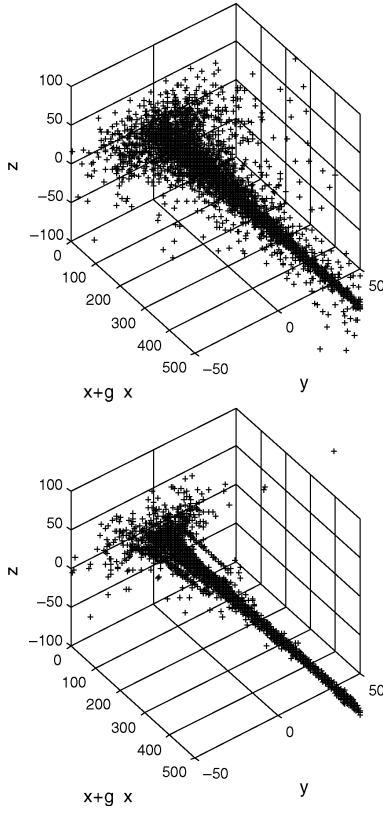


Fig. 6. Particle positions in 4-D for c-PSO and v-PSO, respectively.

the swarm. This situation may result from local or global convergence. Therefore, it may or may not be beneficial to the swarm.

5) *Periodic Localization*: We can intentionally generate local diversity by using an additional vibrational mutation operator, which is called a local mutation operator for a better local search. The application can be done in accordance with the following equation:

$$\begin{aligned}
 x_{i,j}(t) &= x_j^e(t)[1 + A'(0.5 - \text{rand})\delta] \\
 i &= 1, 2, \dots, q \\
 j &= 1, 2, \dots, d \\
 \delta &= \begin{cases} 1 & \text{if } t = nfr' \quad n = 1, 2, \dots \\ 0 & \text{if } t \neq nfr' \end{cases} \quad (45)
 \end{aligned}$$

where A' is a user-defined amplitude factor, fr' is the application frequency, the base vector x^e is the global best called elite particle of the swarm, and q is the number of new particles locally generated by a new vibrational mutation operator.

6) *Multifrequency Vibrational Mutation Applications*: It should be pointed out that (45) is different from (41). The first one is globally, and the second one is locally applied. It is possible to apply both mutation operators by selecting different frequencies and amplitudes. We can describe this situation as

$$\begin{aligned}
 x_{i,j}(t) &= x_{i,j}(t)[1 + A_1(0.5 - \text{rand})\delta] \\
 i &= 1, 2, \dots, s \\
 j &= 1, 2, \dots, d \\
 \delta &= \begin{cases} 1 & \text{if } t = nfr_1 \quad n = 1, 2, \dots \\ 0 & \text{if } t \neq nfr_1 \end{cases} \quad (46)
 \end{aligned}$$

$$\begin{aligned}
 x_{i,j}(t) &= x_j^e(t)[1 + A_2(0.5 - \text{rand})\delta] \\
 i &= 1, 2, \dots, q \\
 j &= 1, 2, \dots, d \\
 \delta &= \begin{cases} 1 & \text{if } t = nfr_2 \quad n = 1, 2, \dots \\ 0 & \text{if } t \neq nfr_2 \end{cases} \quad (47)
 \end{aligned}$$

We can figure the swarm in terms of generations. Assume that fr_1 is equal to 5 and fr_2 is equal to 3. The first seven generations include the following swarm combinations:

t	1	2	3	
S	$S_I^i ^{i=1,2,\dots,s}$	$S_U^i ^{i=1,2,\dots,s}$	$S_U^i ^{i=1,2,\dots,s-q}$ $S_{ML}^i ^{i=1,2,\dots,q}$	
t	4	5	6	7
S	$S_U^i ^{i=1,2,\dots,s}$	$S_{MG}^i ^{i=1,2,\dots,s}$	$S_U^i ^{i=1,2,\dots,s-q}$ $S_{ML}^i ^{i=1,2,\dots,q}$	$S_U^i ^{i=1,2,\dots,s}$

where S_I is the initial swarm, S_U is the swarm that contains the particles generated by update operations, S_{ML} is the swarm that contains the particles locally generated by the second mutation operator given in (47), and S_{MG} is the swarm that contains the particles globally generated by the first mutation operator given in (46). First, the swarm is composed of randomly generated initial particles. In the second generation, the swarm only includes the particles generated by the update operations described in (30). The third generation corresponds to the second frequency $t \leftarrow nfr_2, n = 1$. In the third generation, the swarm includes $(s-q)$ particles generated by update operations and q particles generated by the second vibrational mutation operator. In the fourth generation, the swarm only includes the particles generated by update operations. The fifth generation corresponds to the first frequency $t \leftarrow nfr_1, n = 1$. In the fifth generation, the swarm only includes the particles generated by the first vibrational mutation operator. The sixth generation corresponds to the second frequency $t \leftarrow nfr_2, n = 2$. In the sixth generation, the swarm includes $(s-q)$ particles generated by update operations and q particles generated by the second vibrational mutation operator. In the seventh generation, the swarm only includes the particles generated by update operations. This periodic process goes on until the stopping criterion is satisfied.

Depending on the application options, we can describe two different vibrational PSO algorithms. If we use only the first mutation operator given in (46), we name this type of algorithm a vibrational PSO (v-PSO). If we use both mutation operators given in (46) and (47), we name it a multifrequency vibrational PSO (mv-PSO).

7) *Random and Controlled Diversity*: Vibrational mutation operators generate new particles and these new particles are absolutely random-based estimations for better solutions. Application of the first mutation operator provides a global but random diversity within the swarm. Application of the second mutation operator causes a local but random diversity surrounded by an elite particle neighborhood in the swarm. However, we may improve the quality of estimations generated

TABLE III
DEFINITION OF TEST FUNCTIONS

f	Test Function	$f(x)$	Search Range	x^*	$f(x^*)$
f_1	Ackley	$-20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right)$	$[-30, 30]^D$	$[0, 0, \dots, 0]$	0
f_2	Cosine mixture	$\sum_{i=1}^n x_i^2 - 0.1 \sum_{i=1}^n \cos(5\pi x_i)$	$[-1, 1]^D$	$[0, 0, \dots, 0]$	$-0.1n$
f_3	Exponential	$-\exp \left(-0.5 \sum_{i=1}^n x_i^2 \right)$	$[-1, 1]^D$	$[0, 0, \dots, 0]$	-1
f_4	Griewank	$1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right)$	$[-600, 600]^D$	$[0, 0, \dots, 0]$	0
f_5	Rastrigin	$10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]$	$[-5.12, 5.12]^D$	$[0, 0, \dots, 0]$	0
f_6	Schwefel	$418.9829n - \sum_{i=1}^n x_i \sin \left(\sqrt{ x_i } \right)$	$[-500, 500]^D$	$[420.9687, \dots, 420.9687]$	0

by vibrational mutation operators. We can call this secondary process filtration or controlled diversity. Control action on mutated particles before placing them into the next swarm can be applied by using artificial intelligent methods, such as radial basis function (RBF) neural networks [24], [25].

8) *RBF Neural Networks in Controlled Diversity*: The control process can be applied in global or local diversity applications. Therefore, we have four options: global controlled diversity, local controlled diversity, global random diversity, and local random diversity. Use of global random diversity and local controlled diversity is preferred in this paper. The steps of neural network coupled mv-PSO for optimization are the following. First, cost function calculations in the current swarm are performed to get their objective function values. Second, networks are trained following the threshold generation point by using the particles in the current and previous swarms and their cost function values. However, the training set is limited to a certain number. For this training, particles in the swarms are used as the input and their cost function values are used as the output. Then, mutations according to (45) are applied to the elite particle, and temporal neural network swarm including T_s particles is generated. The objective function values of these particles (f^{NN}) are estimated by using neural nets ($NN_{func}(T_s)$). The particles are sorted in ascending order depending on the objective function values. The best q particles of T_s are randomly placed into the new swarm to be used as candidates at the next step of the algorithm as follows:

$$\begin{aligned}
 f^{NN} &= NN_{func}(T_s) \\
 [f^{NN} \text{ order}] &= \text{sort}(f^{NN}) \\
 x_k(t) &= T_s(\text{order}(i)) \\
 k &= \text{rand}[1 - s] \quad i = 1, 2, \dots, q.
 \end{aligned} \tag{48}$$

There are some important points that require careful tuning during the online use of the neural network model in mv-PSO. These are the number of generations required for training the neural network and the selection of generations to be used for training. The first issue is related to the training cost, and the second one determines the quality of prediction. Instead of gathering all previous exact evaluations, it is more effective to gather a certain amount of collected data (N) based on the recent generations to avoid the computationally expensive task [26]. Additionally, the selection of the neighboring patterns, such as the last certain generations, provides better reliability. The reason for this is that the last certain generations are locally closer in terms of Euclidean distances

in the design space to the current particles and their perturbed variants.

V. APPLICATIONS AND RESULTS

First, v-PSO and the comparative PSO algorithms are tested using test functions, including unimodal and multimodal benchmark functions. Second, mv-PSO algorithm and the comparative PSO algorithms including v-PSO are applied to the shock wave reduction problem in 2-D aerodynamic optimization. In the second comparative study, the low-level flow solver is used to facilitate the computational cost. However, the new methodology is also applied to the same problem by using the high level flow solver.

A. Comparative Results Using Test Functions

The selected test bed is given in Table III. The test procedure includes two different bundles. The first bundle contains the algorithm sensitivity to different function dimensions as D is equal to 10, 20, and 30 with fixed generation size as 10 000 and fixed swarm size as 20. The second bundle contains the behaviors of the algorithms related to different swarm sizes, such as 10, 20, and 30 with fixed dimensions as D is equal to 30, and fixed function evaluations of 200 000. All algorithms are run 100 times and the results are averaged. Peculiar settings are the following: c_1 and c_2 are equal to 2.05 for c-PSO, c_1 and c_2 are equal to 2, w_{ini} and w_{end} are equal to 0.6, 0.2, respectively, for w-PSO, c_1 and c_2 are equal to 2, w_{ini} and w_{end} are equal to 0.6, 0.2, respectively, D_{low} is equal to 0.5 for g-PSO, c is equal to 1.49445, w_{ini} and w_{end} are equal to 0.9, 0.4, respectively, for cl-PSO, c_1 is equal to 1.5, c_2 is equal to 2, w is equal to 0.05, fr is equal to 10, A is equal to 1, and elite count is equal to 3 for v-PSO.

B. Fixed Iteration Rate-Fixed Swarm Size

The resulting values are shown in Table IV. The performance of four algorithms with different dimensions is tabulated in terms of means and averaged CPU times for six test functions. The test runs were executed on an off-the-shelf laptop computer that has an Intel Centrino Duo processor with 64-bit accuracy. The mean value is calculated in accordance with the 95% confidence interval ratio. The best results among the four algorithms are shown in bold. The averaged global best individual values versus generations for D are equal to 30, as shown in Fig. 7. The best performance belongs to v-PSO in almost all test cases. The accuracy and

TABLE IV
FIRST TEST BUNDLE RESULTS

f	D	c-PSO		w-PSO		g-PSO		cl-PSO		v-PSO	
		Mean	t_{CPU}	Mean	t_{CPU}	Mean	t_{CPU}	Mean	t_{CPU}	Mean	t_{CPU}
f_1	10	0.1040±0.0659	1.612	2.5e-15±9.9e-17	1.5219	3.258e-4±4.5e-5	1.634	0.186±0.077	27.49	1.84e-15±2.9e-16	1.717
	20	1.7684±0.2544	1.964	0.0231±0.0322	1.7922	0.0013±1.8e-4	1.95	1.458±0.475	51.33	2.84e-15±1.5e-16	1.957
	30	3.8334±0.3806	2.406	0.0266±0.0374	2.163	0.0027±3.7e-4	2.307	0.972±0.286	76.59	4.93e-15±3.4e-16	2.296
f_2	10	-0.9483±0.0158	1.557	-0.9970±0.0041	1.4688	-0.9998±3.1e-5	1.568	-1±0.21e-15	28.90	-1±0	1.644
	20	-1.533±0.0485	1.938	-1.9512±0.0162	1.74	-1.9979±2.1e-4	1.883	-2±0.902e-9	51.65	-2±0	1.921
	30	-2.0261±0.0692	2.318	-2.8167±0.0286	2.11	-2.9942±5.1e-4	2.255	-3±0.468e-6	94.12	-3±0	2.26
f_3	10	-1±2.2e-18	1.468	-1±0	1.371	-1±1.5e-6	1.422	-1±1.82e-17	26.53	-1±0	1.55
	20	-1±2.15e-17	1.78	-1±1.7e-17	1.593	-0.9999±9.2e-6	1.625	-1±1.62e-11	50.86	-1±3e-18	1.771
	30	-1±5.14e-14	2.136	-1±5.4e-18	1.877	-0.9998±1.9e-5	1.901	-1±0.81e-8	75.31	-1±1e-17	2.039
f_4	10	0.0915±0.0093	2.174	0.0742±0.0075	2.075	0.0549±0.0046	2.125	0.020±0.002	27.89	0.020±0.006	2.251
	20	0.0586±0.0478	2.577	0.0291±0.005	2.396	0.0222±0.0041	2.483	0.0059±0.001	52.67	0.0026±0.002	2.553
	30	0.2238±0.1616	3.043	0.0153±0.0035	2.835	0.0166±0.004	2.946	0.0156±0.002	77.76	8.8568e-4±0.001	2.991
f_5	10	8.6760±0.7852	2.133	5.1837±0.5259	2.055	0.0398±0.0389	2.111	1.19e-9±7.37e-10	27.26	0±0	2.21
	20	38.4849±2.1134	2.635	23.7496±1.6545	2.483	2.8259±0.6476	2.537	5.2±0.496	53.65	0±0	2.621
	30	84.6011±4.6683	3.146	53.5088±2.5935	3.004	14.1188±2.24	3.033	36.76±1.16	76.21	5.6843e-16±1e-15	3.068
f_6	10	836.04±61.8981	2.732	842.7311±59.72	2.185	815.1561±64.3	2.232	2.78±3.44	26.11	620.8131±50.4	2.371
	20	2.375e+3±112.69	4.222	2.1577e+3±101	3.093	2.1798e+3±120.9	3.021	1.69e3±59.98	51.78	1.3384e+3±68.5	3.128
	30	4.146e+3±177.38	5.451	3.62e+3±158.35	3.945	3.47e+3±149.3	3.781	4.67e3±93.70	77.31	2.1395e+3±103.3	3.971

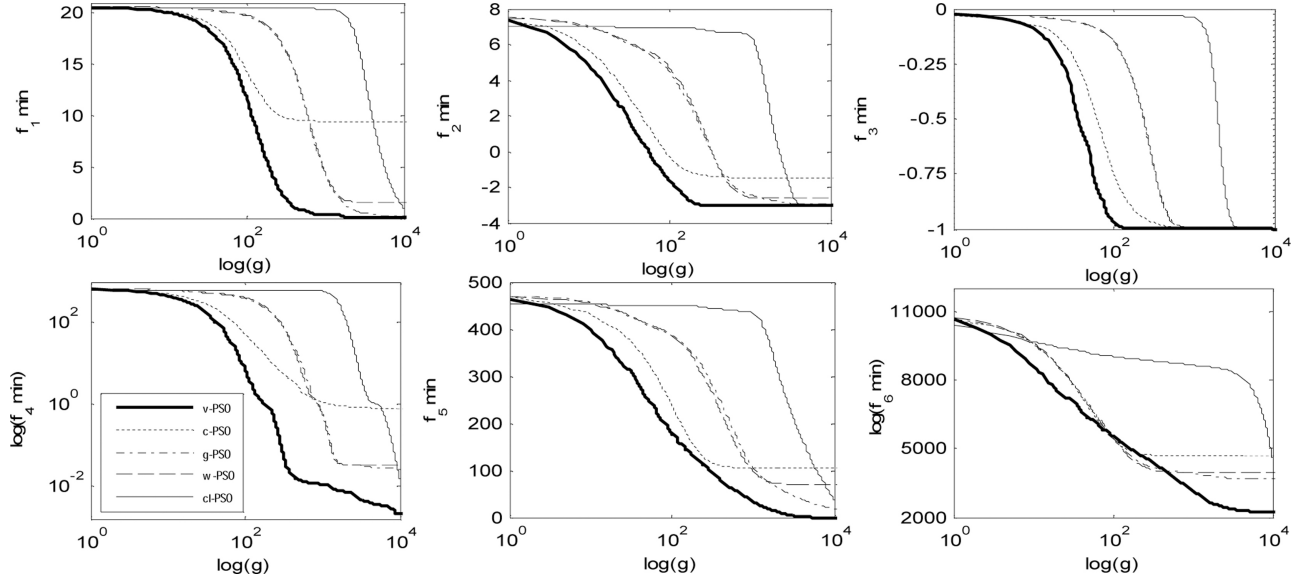


Fig. 7. Averaged best function values versus generations for test functions.

the efficiency superiorities of v-PSO can be clearly seen in Fig. 7. It converges within 10^2 or 10^3 generation number to the acceptable objective function values. However, the other algorithms converge within 10^4 generation number to the same or worse objective function values.

C. Fixed Function Evaluation Rate-Fixed Dimension Rate

The averaged best function values versus swarm sizes that are 10, 20, and 30 for D that is equal to 30 are shown in Fig. 8 and Table V. The mean value is calculated in accordance with the 95% confidence interval ratio. The best results among the four algorithms are shown in bold. Similar results are observed in comparison with the previous analysis. v-PSO has the best performance in all function optimizations. Fig. 8 shows an interesting result in the relationship between the performance and the swarm size. Generally, v-PSO has

similar performances for all swarm sizes. This means v-PSO provides a better efficiency in lower swarm sizes. However, the other algorithms' performances, with the exception of cl-PSO, significantly depend on the swarm size. Typically, larger swarm size means better performance for them. The main reason for this result is that v-PSO provides enough diversity within the swarm even in a low swarm size. The other algorithms need a larger swarm size to have enough diversity in the swarm. On the other hand, the main drawback of cl-PSO seems to be a slow convergence rate. It needs more iteration numbers rather than a larger population size to get more accurate results.

D. Shock Wave Reduction Problem

Aerodynamic shape optimization is a rewarding field for numerical analysis since it poses several challenges due to

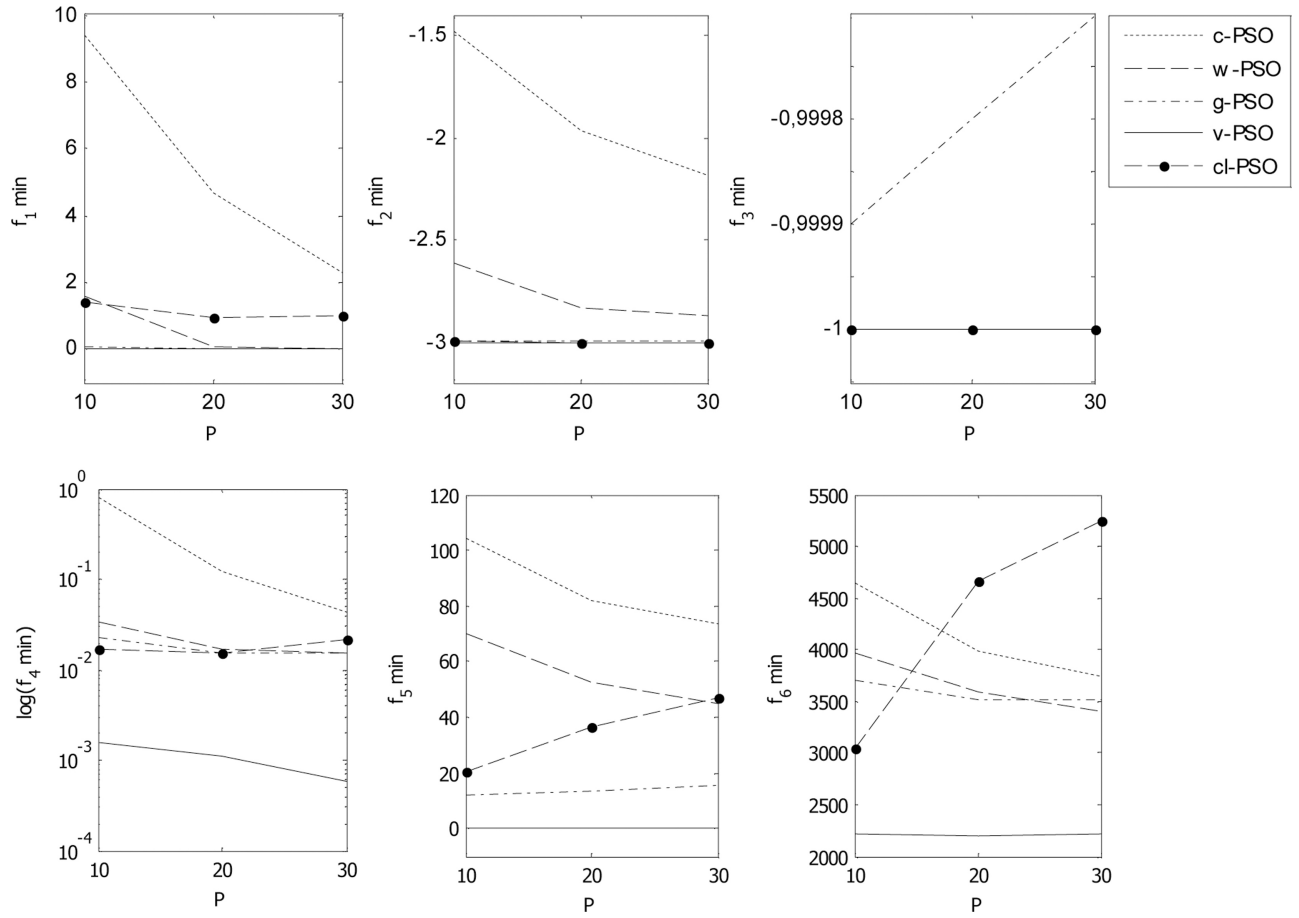


Fig. 8. Averaged best function values versus swarm sizes for selected test functions.

the complexity of the subproblems involved. In aerodynamic shape optimization, the goal is to improve the aerodynamic performance of the design given by a set of parameters. The improvement is measured by the reduction of a cost function, which typically is an expression based on the drag (C_D), the lift (C_L), and the moment (C_M) coefficients. Aircraft wings that contribute the most lift and drag, and determine as much as 30% weight of the whole aircraft, are the primary subject for optimization efforts. Airfoils are the basic elements of wing geometry. They determine a large share of wing flow phenomena though they are just 2-D sections of the physical wing surface.

In transonic flow conditions, the shock waves occur on the airfoil surfaces. These shock waves decrease lift and cause an increase in drag. Therefore, they are undesirable phenomena in transonic flow regimes. Several different search methods are studied to solve shock wave reduction problems besides gradient-based algorithms, such as simulated annealing, ant colony optimization, artificial neural networks, and GAs. Another approach that is gaining popularity is PSO algorithm. In recent studies, PSO algorithms have been applied to design wing in structural and aerodynamical objectives. Tan *et al.* [27] demonstrated the fast convergence capabilities of both the swarm and the EA of airfoil shape optimization problems. In his comparative study, the EA exhibited marginally higher efficiency over the swarm for single-objective airfoil design problems, while the swarm performed better on the multiobjec-

tive examples. Ng *et al.* [28] and Ray *et al.* [29] demonstrated that the PSO algorithm is capable of handling different forms of airfoil shape optimization problems. Venter *et al.* [30], [31] presented that the particle swarm optimization algorithm is able to reliably find the optimum design, and it is capable of dealing with the unique challenges posed by multidisciplinary optimization as well as the numerical noise and truly discrete variables present in the optimization of a typical transport aircraft wing. An example of using parallel computing in a synchronous and asynchronous manner was also presented by Venter *et al.* Khuranal *et al.* [32] showed that particle swarm optimization and artificial neural networks can be coupled in a surrogate model approach. However, these studies also showed that PSO algorithm needs to be more efficient. Our improved algorithms are used to solve shock wave reduction problems of NACA64A410 and NACA64A010 airfoils.

E. Airfoil Representation

An airfoil curve can be represented by two Bezier curves [33] including upper and lower curves, and each Bezier curve is governed by m control points. In our case, m was selected as 12. The Bezier curve is described as follows:

$$\begin{aligned} x(u) &= \sum_{i=0}^m C_m^i u^i (1-u)^{m-i} x_i \\ y(u) &= \sum_{i=0}^m C_m^i u^i (1-u)^{m-i} y_i \\ C_m^i &= \frac{m!}{i!(m-i)!} \quad 0 \leq u \leq 1 \end{aligned} \quad (49)$$

TABLE V
SECOND TEST BUNDLE RESULTS

f	P	c-PSO	w-PSO	g-PSO	cl-PSO	v-PSO
		Mean	Mean	Mean	Mean	Mean
f_1	10	9.412±0.6149	1.5913±0.4344	0.0803±0.096	1.4070±0.4721	4.583e-15±3e-16
	20	4.6684±0.4644	0.0902±0.067	0.0026±3.7e-4	0.9717±0.2864	4.9738e-15±3e-16
	30	2.2994±0.2776	0.0116±0.023	0.0027±3.9e-4	0.9990±0.2390	5.0449e-15±3e-16
f_2	10	-1.4844±0.0918	-2.6113±0.05	-2.996±3.3e-4	-2.9985±0.0029	-3±0
	20	-1.9673±0.0767	-2.833±0.03	-2.9937±6.1e-4	-3±0.4684e-06	-3±0
	30	-2.1842±0.071	-2.8714±0.03	-2.9928±7.3e-4	-3±8.0217e-07	-3±0
f_3	10	-1±8.8e-5	-1±1e-17	-0.9999±1e-5	-1±8.2583e-09	-1±1e-17
	20	-1±2e-13	-1±3e-18	-0.9998±1.74	-1±0.8138e-08	-1±1e-17
	30	-1±1e-16	-1±2e-18	-0.9997±2e-5	-1±1.5968e-08	-1±1e-17
f_4	10	0.8154±0.275	0.0339±0.01	0.0223±0.005	0.0165±0.00210	0.0016±0.002
	20	0.1233±0.064	0.0167±0.003	0.0155±0.003	0.0156±0.00207	0.0011±0.001
	30	0.0431±0.01	0.0153±0.003	0.0153±0.003	0.0216±0.00190	5.9021e-4±0.001
f_5	10	104.3609±5.25	69.9753±3.63	12.1629±2.31	20.1261±1.1369	0±0
	20	81.8152±3.98	52.8919±2.42	13.5722±1.77	36.7680±1.1605	1.1369e-15±1e-15
	30	73.6367±3.7	44.8626±2.45	15.4020±1.98	46.7153±1.2831	5.6843e-16±1e-15
f_6	10	4.6531e+3±142	3.9629e+3±161	3.6985e+3±194	3.048e+3±121.8	2.2246e+3±123
	20	3.978e+3±149	3.5837e+3±147	3.5119e+3±188	4.672e+3±93.7	2.1888e+3±110
	30	3.7372e+3±140	3.4045e+3±134	3.5136e+3±152	5.2401e+3±78.7	2.2258e+3±107

where u is the parameter of the curve whose values vary uniformly between $[0, 1]$, x_i , y_i are the coordinates of the control points that define the profile coordinates $x(u)$, $y(u)$. The two control points $(0, 0)$ and $(1, 0)$ at the leading and trailing edges are fixed. Normally, the x_i coordinates are fixed so that the design parameters are only the y_i values. Therefore, a total of 22 design parameters for the upper and lower curves are taken to be optimized. Initial swarms are generated by using a random number operator based on NACA64A410 airfoil control points.

F. Objective Function

The objective function value is based on C_D and C_L coefficient ratios. The airfoil maximum thickness is bounded to a design target value, t^* . The objective function f is given as follows:

$$f = \left[\frac{C_D}{C_L} + 10(C_L^* - C_{L2})^2 + 100(t^* - t)^2 \right] \quad (50)$$

$$C_{L2} = \begin{cases} C_L^* & \text{if } C_L \geq C_L^* \\ C_L & \text{if } C_L < C_L^* \end{cases}$$

where C_L^* is the design lift coefficient.

G. Methodology

The baseline airfoil shape is optimized in accordance with given flow conditions by using six PSO algorithms, including c-PSO, w-PSO, g-PSO, cl-PSO, v-PSO, and mv-PSO. The swarm size is selected as 10; the maximum generation number is selected as 100. The problem dimension is fixed to 22 as the control points of Bezier curves. The other features peculiar to the algorithms are given in Table VI. In a comparative study, all algorithms are run 20 times and the averaged global best particle values versus computational fluid dynamics (CFD) calls are taken into consideration for a fair comparison. Additionally, the schematic diagram of mv-PSO is depicted in Fig. 9.

TABLE VI
FEATURES OF PSO ALGORITHMS

Algorithm	c_1	c_2	c			
c-PSO	2.05	2.05	–			
w-PSO	2	2	–	w		
				0.6–0.2		
cl-PSO	–	–	1.49445	0.9–0.4		
g-PSO	2	2	–	d_{low}		
				10^{-3}		
				fr		
				A_1		
v-PSO & mv-PSO	2	2	–	0.6–0.2	fr_1 15	0.5
					fr_2 2	A_2
						0.5

In mv-PSO applications, two networks are trained after the threshold generation point (in the implementation, it is selected as the tenth generation) by using the particles in the current and previous swarms and their cost function values. The first network is trained for C_L estimation. The second network is trained for C_D estimation. However, the training set is limited to a certain number (in the implementation, the training set is limited with the last 100 data set, which means the last tenth generations).

H. Grid Generation and Flow Solver in the Comparative Study

The flow around an airfoil is assumed to be inviscid compressible flow. The pressure distribution around the airfoil can be determined by solving, for example, Euler equations presented as follows:

$$\nabla(\rho u U) = -\frac{\partial p}{\partial x} \quad (51)$$

$$\nabla(\rho v U) = -\frac{\partial p}{\partial y}$$

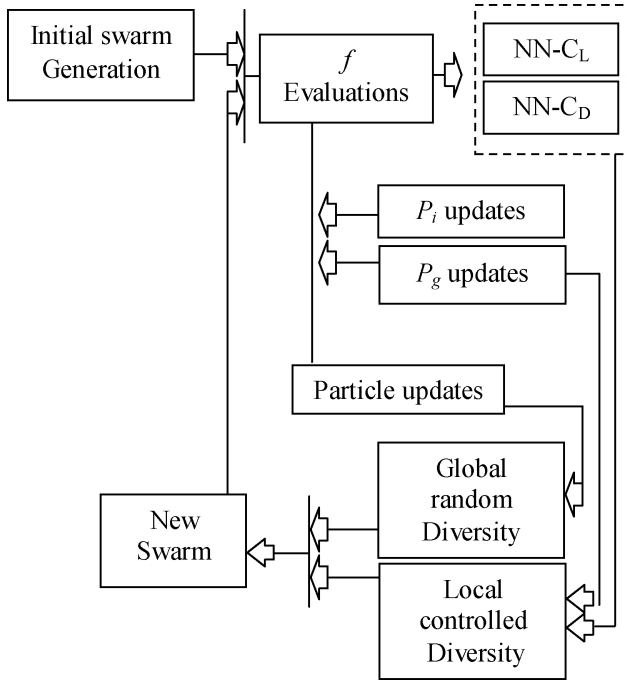


Fig. 9. Flow chart of mv-PSO.

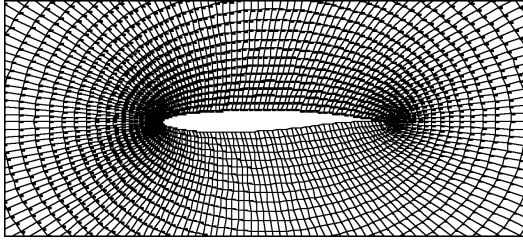


Fig. 10. Structured O-mesh grid around NACA64A410.

where ρ is the fluid density, p is the pressure, u is the flow speed in the x -direction, v is the flow speed in the y -direction, and $\mathbf{U}$ is the flow velocity vector. The given equations are solved using a finite volume method and the Roe flux-splitting scheme [34]. The elliptic partial differential equation solution method is used to generate a structured O-mesh grid domain given in Fig. 10. The airfoil is placed at 2° of attack and the flow has a *Mach* number of 0.75.

I. Comparative Results

The optimization results are depicted in Fig. 11. In this figure, the averaged global best particle objective function values and CFD calls are plotted. Among the classical PSO algorithms including c-PSO, w-PSO, cl-PSO, and g-PSO, the best performance belongs to g-PSO algorithm. It reaches the value of 24.08 in terms of $1/\text{averaged } f_{\min}$ at 100th generation, which means 1000 CFD calls. c-PSO is the poorest algorithm reaching the value of 22.13 at 100th generation. c-PSO seriously suffered from a lack of diversity and was trapped by a local optimum. The result that belongs to cl-PSO also shows that the comprehensive learning PSO needs more iterations to get better results. Similar to previous results, it seems that it suffers from a slow convergence rate. The results demonstrate that the classical algorithms, with the

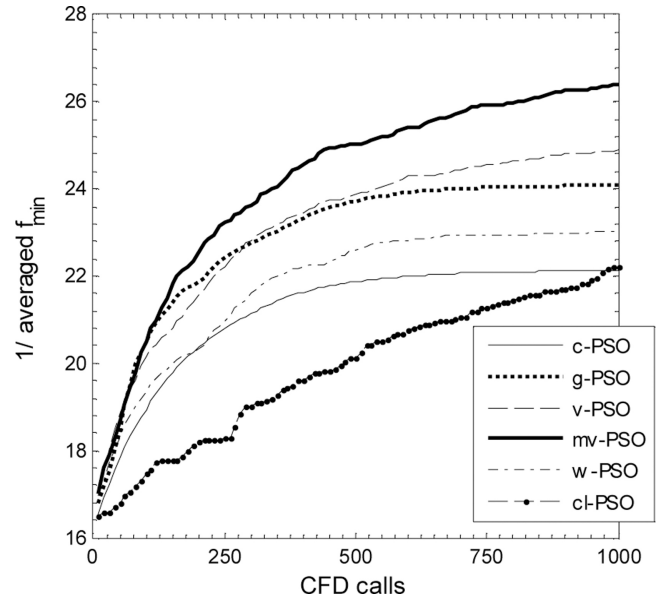


Fig. 11. Optimization results for PSO algorithms.

exception of cl-PSO, are unable to converge to a better solution.

On the other hand, v-PSO does outperform the regular algorithms. It reaches the value of 24.1 at 56th generation, which means 560 CFD calls. This result also means a 44% decrease in the required generations as compared with g-PSO. At the end of the optimization process, v-PSO reaches the value of 24.89. mv-PSO does outperform all algorithms. After the 13th generation, the controlled diversity tool shows its effects on the optimization process. This is an expected situation because the activation of neural nets starts with the tenth generation. After this threshold generation, the local controlled diversity tool places new particles into a swarm and these particles accelerate the optimization process. The random global diversity tool also provides a global search capability to the algorithm. mv-PSO reaches the value of 24.19 at 36th generation. This means a 64% decrease in the required CFD calls as compared with g-PSO. Additionally, it reaches the value of 24.89 at 44th generation, which means 440 CFD calls, while v-PSO reaches the same value at 1000 CFD calls. mv-PSO provides the value of 26.39 at 100th generation.

In Table VII, the original and an example of optimized airfoil features are presented. According to the Euler solver results, the optimization process keeps the lift coefficient and maximum thickness ratio almost fixed while decreasing the drag coefficient by 40%. This results in a 65% increase in the C_L/C_D ratio.

In Fig. 12, typical pressure coefficient distributions and the airfoil shapes formed by the optimization process are represented. According to these figures, the shock wave occurred on the upper surface of the airfoil is almost eliminated by the optimization process.

J. Verification Study

In the second study, the same problem with different airfoil and flow conditions is solved by only mv-PSO algorithm. A

TABLE VII
AERODYNAMIC FEATURE COMPARISONS IN ACCORDANCE
WITH EULER SOLVER

	Euler Solver			
	C_L	C_D	C_L/C_D	T
Original	0.8847	0.0578	15.3062	0.1
Optimized	0.8860	0.035	25.31	0.998

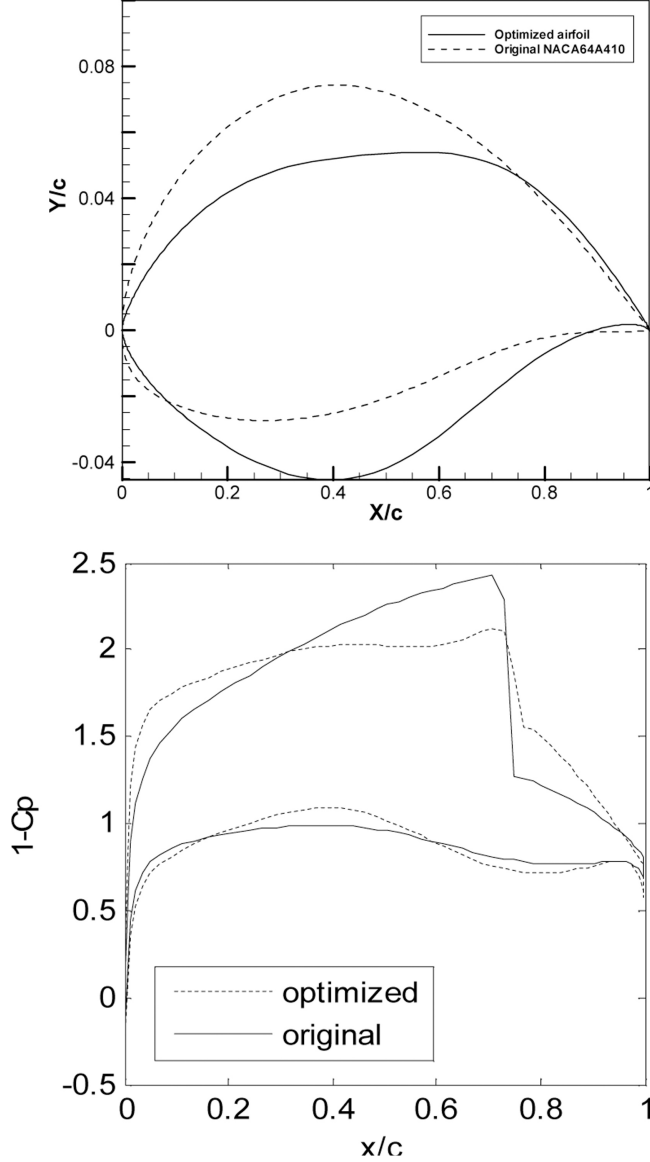


Fig. 12. Original and optimized airfoil shapes and pressure coefficient distributions.

well-known Navier–Stokes solver [35] is used to determine aerodynamic features around the airfoil.

K. Grid Generation and Flow Solver

Ten percent thick symmetric NACA 64A010 was used as the verification airfoil. The resolution of the utilized C-type computational grid is 449×121 . Normal spacing for the first grid line of the surface of the airfoil was $0.00001c$. In the normal direction, 121 grid points were utilized with the far field boundary located at a distance equal to 13 airfoil chord

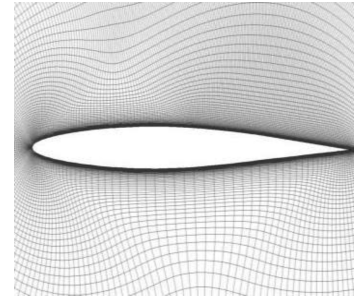


Fig. 13. Computational grid used in simulations.

TABLE VIII
GRID SENSITIVITY FOR NACA 64A010 AIRFOIL

Grid Size	C_L	C_D	C_{FD}	C_{WD}
449 * 121	0.13	0.0329	0.005540	0.027338
389 * 121	0.1357	0.03233	0.00544	0.026889
281 * 99	0.1298	0.03261	0.005452	0.027158
Comp. [33]	0.14	0.0328	0.00544	0.0274

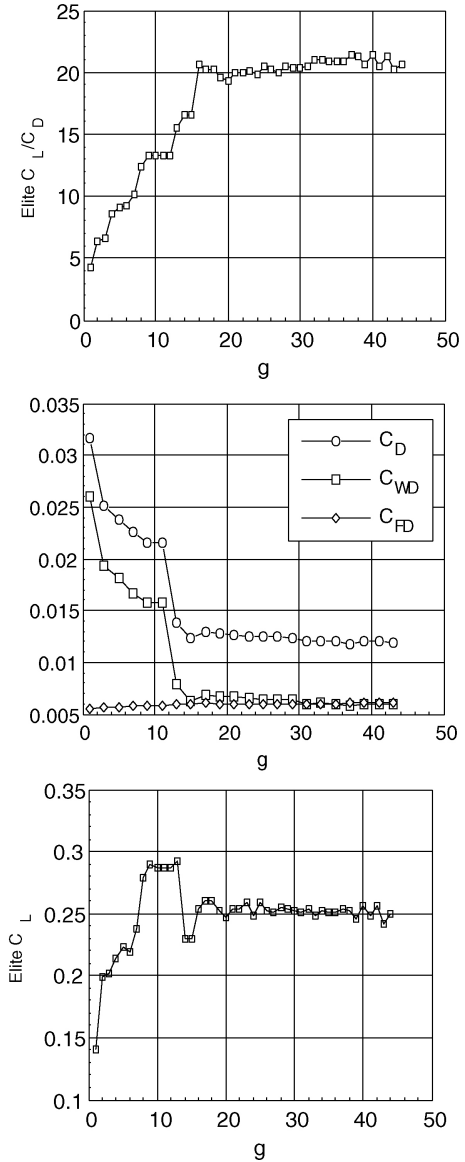


Fig. 14. Change of aerodynamic coefficients and performance during the generations.

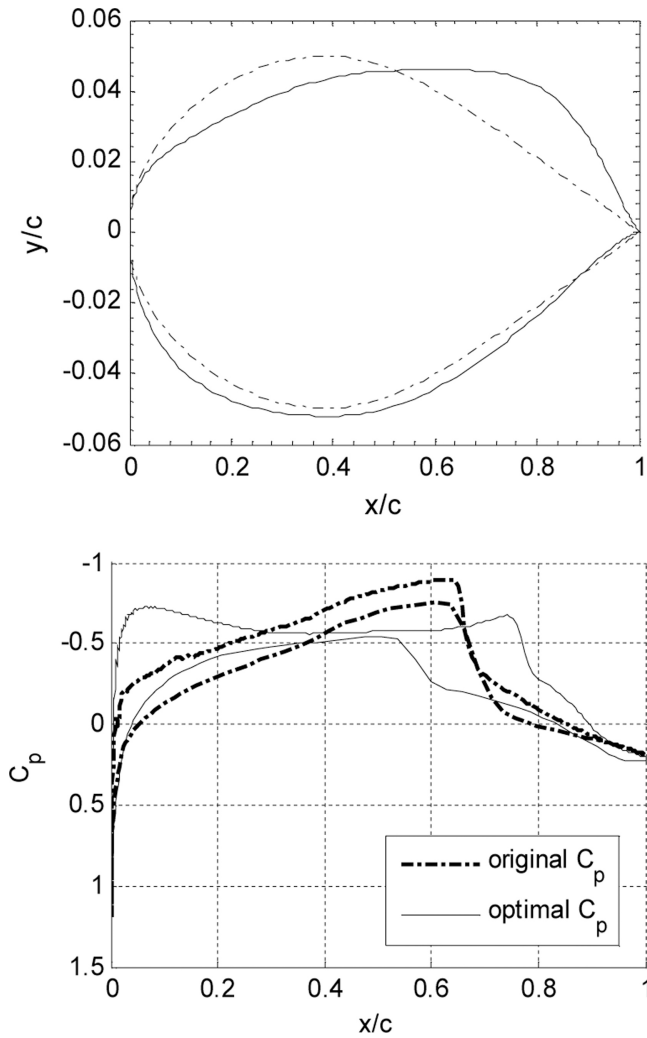


Fig. 15. Original and optimized airfoil shapes and pressure coefficient distributions.

(c) lengths. The grid used in the simulations is shown in Fig. 13. The grid is divided into four sub-blocks to implement the parallel computing for a faster computation. The validation case used in this paper had a Reynolds number of 6.07 million based on airfoil chord, Mach number 0.85, and angle of attack 1.0° corresponding to one of the numerical studies. Under the above free stream conditions, the predicted sectional lift and drag are equal to 0.13 and 0.0329, respectively. The predicted components of the airfoil drag are 0.27338 representing the pressure component (C_{WD}) corresponding to 83% of the total drag, and 0.00554 representing the skin friction component (C_{FD}) corresponding to 17% of the total drag. Solution sensitivity to the grid used in this paper is illustrated in Table VIII. For the computational cases, three sets of grids and another computational result given by Hassan *et al.* [36] have been used. Normalized first cell height y^+ values, based on the height of the first wall-bounded cell, are below unity for all the meshes considered here. The solutions obtained on the course and fine grids are reasonably good. The results obtained on these three different grid sizes are reasonably grid-converged results, and prove little solution sensitivity.

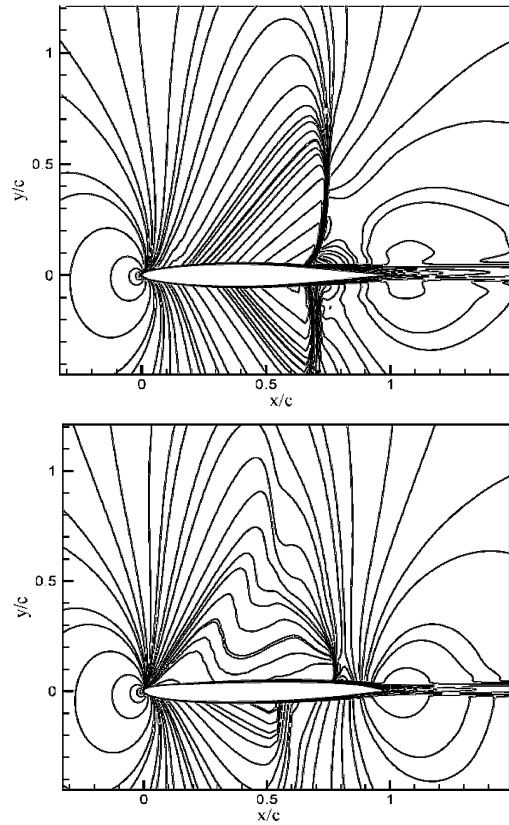


Fig. 16. Mach number contours around original and optimized airfoils.

A Reynolds averaged Navier–Stokes (RANS) CFD code was used to perform the computational simulations. The thin-layer approximation to the time-dependent, conservation form of the RANS equations is solved by an implicit, approximately factored, finite volume, upwind and multigrid algorithm. Due to the high Reynolds number of the flow, it was assumed to be fully turbulent. The one-equation Spalart–Allmaras model was used to compute a turbulent eddy viscosity. The details of the code can be found in Krist *et al.* [35]. A two-level multigrid technique was used to achieve the convergence acceleration.

L. Results

The optimization process takes about 40 generations. The changes of aerodynamic coefficients C_L , C_D , and the aerodynamic performance f versus generations that belong to the best individual are depicted in Fig. 14. After forming at optimal values, C_L increases by 86%. On the other hand, C_D decreases by 63.5%. The total drag decrement comes from wave drag rather than from drag. In the end, C_{FD} takes the value of 0.0060 and C_{WD} takes the value of 0.0059, which is less than wave drag. As a result, the objective function f increases by 412.76%.

The resulting shape and its C_p distribution are shown in Fig. 15. The maximum thickness ratio for the optimal airfoil form is about 0.0963, which is good enough for the process. During the optimization, the crescent point for the upper surface moved toward the trailing edge. This movement results in the relocation of the shock waves. The upper shock wave moved from the original place $0.695c$ toward $0.782c$.

The lower shock wave is also moved toward the leading edge. Both strengths of the shock waves are significantly decreased. Fig. 16 shows the Mach number contours around the original and optimized airfoil shapes. While strong shock waves are observed on both the upper and lower surfaces of the NACA 64A010 airfoil, the optimized airfoil form weakens the strong shocks, which are clearly depicted on the upper and lower sides.

VI. CONCLUSION

We observed that in the general case a particle surfs on sine waves in nature. A particle seeking an optimal location attempted to catch another wave randomly, manipulating its frequency and amplitude. However, manipulations were not enough to catch the big wave directing to the global solution. During the optimization process, the swarm lost its diversity and the particles became more and more similar to each other, and gathered into the neighborhood of the best particle in the swarm.

Although mutation applications provided diversity in the swarm, the diversity without continuous impetus disappeared during the generations. A periodic mutation application seemed to help the particle to jump onto another big wave. We obtained the following results.

- 1) The principal role of this multifrequency approach was to answer the question of which particles should be mutated, when they should be mutated, and which ones should be merged into the swarm.
- 2) The new mutation operators, when combined with a mutation strategy, provided local and global diversities during the updating phases of the generations.
- 3) The first mutation operator was applied to all particles of the whole swarm and this application provided global but random diversity in the swarm. The global diversity afforded a chance for the swarm to escape from all local optima.
- 4) The second mutation operator was applied specifically to the variables of the elite particle and this application provided local diversity in the swarm.
- 5) Additionally, the new approach introduced a random diversity and a controlled diversity.
- 6) The control provided by neural nets renders available a local, but controlled diversity, which leads to a fast convergence.

Based on the results obtained, it was concluded that the present vibrational PSO algorithms were efficient and fast.

REFERENCES

- [1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromach. Hum. Sci.*, 1995, pp. 39–43.
- [2] E. Ozcan and C. K. Mohan, "Particle swarm optimization: Surfing the waves," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1939–1944.
- [3] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
- [4] Y. Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.
- [5] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. 7th Ann. Conf. Evol. Prog.*, 1998, pp. 591–601.
- [6] H. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Proc. IEEE Swarm Intell. Symp.*, Apr. 2003, pp. 72–79.
- [7] A. Stacey, M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," in *Proc. IEEE Congr. Evol. Comput.*, Dec. 2003, pp. 1425–1430.
- [8] M. Pant, R. Thangaraj, V. P. Singh, and A. Abraham, "Particle swarm optimization using Sobol mutation," in *Proc. 1st IEEE Int. Conf. Emerging Trends Eng. Tech.*, Jul. 2008, pp. 367–372.
- [9] A. E. Zavala, A. H. Aguirre, E. R. V. Diharce, and S. B. Rionda, "Constrained optimization with an improved particle swarm optimization algorithm," *Int. J. Intell. Comput. Cybern.*, vol. 1, no. 3, pp. 425–453, 2008.
- [10] D. Jia, L. Li, Y. Zhang, and X. Chen, "Particle swarm optimization combined with chaotic and Gaussian mutation," in *Proc. 6th World Congr. Intell. Cont. Auto.*, Jun. 2006, pp. 3281–3285.
- [11] X. Wu, B. Cheng, J. Cao, and B. Cao, "Particle swarm optimization with normal cloud mutation," in *Proc. 7th World Congr. Intell. Cont. Auto.*, Jun. 2008, pp. 2828–2832.
- [12] J. Chen, Z. Ren, and X. Fan, "Particle swarm optimization with adaptive mutation and its application research in tuning of PID parameters," in *Proc. 1st Int. Symp. Syst. Cont. Aerospace Astronautics*, Jan. 2006, pp. 990–994.
- [13] B. Liu, L. Wang, and Y. H. Jin, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.
- [14] M. Yang, H. Huang, and G. Xiao, "A novel dynamic particle swarm optimization algorithm based on chaotic mutation," in *Proc. 2nd IEEE Int. Workshop Knowl. Discovery Data Mining*, Jan. 2009, pp. 656–659.
- [15] J. Liu, X. Fan, and Z. Qu, "An improved particle swarm optimization with mutation based on similarity," in *Proc. 3rd Int. Conf. Natural Comput.*, 2007, pp. 824–828.
- [16] C. Biao, L. Zhishu, F. Die, H. Jian, O. Peng, and L. Qing, "Mutated fast convergent particle swarm optimization and convergence analysis," in *Proc. IEEE 1st Int. Conf. Intell. Netw. Intell. Syst.*, Nov. 2008, pp. 5–8.
- [17] B. Luitel and G. K. Venayagamoorthy, "A PSO with quantum infusion algorithm for training simultaneous recurrent neural networks," in *Proc. IEEE-INNS IJCNN*, Jun. 2009, pp. 1923–1930.
- [18] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. World Congr. Comput. Intell.*, 1998, pp. 69–73.
- [19] M. Pant, T. Radha, and V. P. Singh, "A new diversity based particle swarm optimization using Gaussian mutation," *Int. J. Math. Modeling Simulation Appl.*, vol. 1, no. 1, pp. 47–60, 2008.
- [20] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
- [21] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi, *Wavelets and Their Applications*. ISTE Ltd., London, U.K., 2007, pp. 8–12.
- [22] P. Abry, *Ondelettes et Turbulence. Multirésolutions, Algorithmes de Décomposition, Invariance d'échelles*. Paris, France: Diderot Editeur, 1997.
- [23] Y. V. Pehlivanoglu, "Hybrid intelligent optimization methods for engineering problems," Ph.D. dissertation, Dept. Aerospace Eng., Old Dominion Univ., Norfolk, VA, 2010.
- [24] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.
- [25] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, 2003.
- [26] M. K. Karakasis and K. C. Giannakoglu, "On the use of metamodel-assisted, multiobjective evolutionary algorithms," *Eng. Opt.*, vol. 38, no. 8, pp. 941–957, 2006.
- [27] C. M. Tan, T. Ray, and H. M. Tsai, "A comparative study of evolutionary algorithm and swarm algorithm for airfoil shape optimization problems," in *Proc. AIAA 41st Aerospace Sci. Meet. Exhibit.*, 2003.
- [28] K. Y. Ng, C. M. Tan, T. Ray, and H. M. Tsai, "Single and multiobjective wing planform and airfoil shape optimization using a swarm algorithm," in *Proc. 41st AIAA Aerospace Sci. Meeting Exhibit.*, 2003.
- [29] T. Ray and H. M. Tsai, "Swarm algorithm for single and multiobjective airfoil design optimization," *AIAA J.*, vol. 42, no. 2, pp. 366–373, 2004.
- [30] G. Venter and J. Sobieski, "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization," in *Proc. 9th AIAA/ISSMO Symp. Multidisciplinary Anal. Optimization*, 2002.
- [31] G. Venter and J. Sobieski, "Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations," *J. Aerospace Comput., Inform. Commun.*, vol. 3, pp. 123–137, Mar. 2006.

- [32] M. S. Khurana, H. Winarto, and A. K. Sinha, "Application of swarm approach and artificial neural networks for airfoil shape optimization," in *Proc. 12th AIAA/ISSMO Multidisciplinary Anal. Optimization Conf.*, 2008.
- [33] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. New York: Academic, 1993, pp. 41–42.
- [34] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Comput. Phys.*, vol. 43, no. 2, pp. 347–372, 1981.
- [35] S. L. Krist, R. T. Biedron, and C. L. Rumsey, *CFL3D User's Manual Version 5.0*, NASA/TM-1998208444, Jun. 1998.
- [36] A. A. Hassan, B. Osborne, S. Schwimley, and G. Billman, "Control of shock-boundary layer interactions (SBLIs) using an oscillatory jet," in *Proc. 45th AIAA Aerospace Sci. Meeting Exhibit.*, Jan. 2007.



Y. Volkan Pehlivanoglu received the B.S. degree from the Department of Aeronautical Engineering, Istanbul Technical University, Istanbul, Turkey, in 1993, the B.S.S. degree from the Department of International Relations, Istanbul University, Istanbul, in 1998, and the M.S. and Ph.D. degrees from the Department of Aerospace Engineering, Old Dominion University, Norfolk, VA, in 2006 and 2010, respectively.

He is currently with the Department of Aerospace Engineering, Turkish Air Force Academy, Yesilyurt, Istanbul. His current research interests include computational fluid dynamics, design, and optimization methods.