A dynamic adaptive PSO based on chaotic search for constrained multiobjective optimization Problems

Abdallah A. Mousa, B. N. Al-Matrafi, Abdallah A. Galal

1Department of Mathematics and Statistics, Faculty of Sciences, Taif University, Taif, KSA
2Department of Basic Engineering Science, Faculty of Engineering, Menoufia University, Shbin Al Kom, Egypt
3Physics and Engineering Mathematics Department, Faculty of Engineering, Tanta University, Tanta, Egypt

Abstract—In this paper, a new enhanced particle swarm optimization method with dynamic adaptation is presented. To enrich the searching behaviour and to avoid being trapped into local optimum, dynamic adaptive of PSO parameters are incorporated into PSO. In this algorithm, inertia weight of each particle is dynamically updated, also personal influence and social influence parameters are dynamically adapted during the process. Moreover, a modified velocity updating formula of the particle is presented, where, a new constriction factor which control the feasibility of the particles is presented. The proposed approach can be viewed as the global optimization algorithm while the chaotic search CS is employed for the local search. Thus, the possibility of exploring a global minimum in problems with many local optima is increased. The proposed method can obtain the global optimal results quickly, due to fast globally converging characteristics of PSO and the effective local search ability of CS. The results, provided by the proposed algorithm for benchmark engineering problems, are promising when compared with exiting well-known algorithms. Also, our results suggest that our algorithm is better applicable for solving real-world application problems.

Index Terms—multiobjective optimization, particle swarm, local search; chaos

1 INTRODUCTION

In multiple objectives optimization, several conflicting objectives have to be minimized simultaneously. Generally, no unique solution exists but a set of mathematically equally good solutions can be identified, by using the concept of Pareto optimality. Many applications of constrained optimization can be found in engineering [1–2], economics and finance [3], medicine [4–5], management and planning [6], etc. There exist many solution strategies to solve the constrained optimization problems. One of the basic approaches is the weighting method [7], where one single-objective optimization problem is formed by weighting several objective functions. Similar problem has the ε-constraint method, introduced in [7].

Recently, there has been a boom in applying evolutionary algorithms to solve multiobjective optimization problems [8–11]. Evolutionary algorithms (EAs) are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social behavior of species. The development of metaheuristic optimization theory has been flourishing. Many metaheuristic paradigms such as genetic algorithm [8,12,13], simulated annealing [14], tabu search [14,15], and ant colony algorithm [16] has become an interesting approach to solve many hard problems. Recently particle swarm optimization PSO [11,17] have shown their efficacy in solving computationally intensive problems. Animals, especially birds, fishes etc. always travel in a group without colliding, each member follows its group, adjust its position and velocity using the group information, because it reduces individual’s effort for search of food, shelter etc. Particle swarm optimization is evolutionary technique similar to genetic algorithm because both are population based and are equally affective. Particle swarm optimization has better computational efficiency, i.e. it requires less memory space and lesser speed of CPU, it has less number of parameters to adjust. Genetic algorithm and other similar techniques (e.g. simulated annealing), work for discrete design variables, whereas particle Swarm optimization work for discrete as well as analogue systems, because it is inherently continuous, does not need D/A or A/D conversion. Although for handling discrete design variables Particle swarm optimization needs some modification to be done in particle swarm optimization methods. PSO is an evolutionary computation technique, developed for optimization of continuous nonlinear, constrained and unconstrained, non differentiable multimodal functions [18]. PSO is inspired firstly by general artificial life, the same as bird flocking, fish schooling and social interaction behaviour of human and secondly by random search methods of evolutionary algorithm [19]. In [20], a dynamic multi-swarm particle swarm optimizer (DMS-PSO) was proposed whose neighborhood topology is dynamic and randomized. DMS-PSO gives a better performance on multimodal problems than some other PSO variants, but the local search performance is not satisfactory. Dynamic pattern are challenging for PSO, a self-adapting multi-swarm has been derived [21]The multi-swarm with exclusion has been favorably compared, on the moving peaks problem, to the hierarchical swarm, PSO re-initialization and a state-of-the-art dynamic optimization evolutionary algorithm known as self-organizing scouts. To deal with discrete events, an algorithm based on discrete developed in [22]. This approach solves the overlapping coalition formation problem in multiple virtual organiza-
2. MULTIOBJECTIVE OPTIMIZATION (MO)

A general optimization (minimization) problem [7] of $M$ objectives can be mathematically stated as:

$$\text{Minimize } f(x) = \left[ f_i(x), i = 1,2,...,M \right]$$

subject to $g_j(x) \leq 0, j = 1, 2,...,J$.

given $x = [x_1, x_2, ..., x_n]$, where $n$ represents the dimension of the decision variable space, $f_i(x)$ is the $i$-th objective function, and $g_j(x)$ is the $j$-th inequality constraint. The MO problem then reduces to finding an $x$ such that $f_i(x)$ is optimized. Since the notion of an optimum solution in MO is different compared to the single objective optimization (SO), the concept of Pareto dominance is used for the evaluation of the solutions.

Definition 1: (Pareto dominance). A vector $\vec{u} = [u_1, u_2, ..., u_M]$ is said to dominate $\vec{v} = [v_1, v_2, ..., v_M]$ ($\vec{u}$ dominates $\vec{v}$ denoted by $\vec{u} \triangleright \vec{v}$), for a MO minimization problem, if and only if

$$\forall i \in \{1,...,M\}, u_i \leq v_i \land \exists i \in \{1,...,M\}: u_i < v_i$$

where $M$ is the dimension of the objective space.

Definition 2: (Pareto optimality). A solution $\vec{u} \in U$, where $U$ is the universe, is said to be Pareto optimal if and only if there exists no other solution $\vec{v} \in U$, such that $\vec{u} \triangleright \vec{v}$. Such solutions $\vec{u}$ are called nondominated solutions. The set of all such nondominated solutions constitutes the Pareto-Optimal Set.

Definition 2: (Ideal objective vector): An objective vector minimizing each of the objective functions is called an ideal (perfect) objective vector. The component $v^i_r$ of the ideal objective vector $\vec{z}^i \in \mathbb{R}^n$ are defined by minimizing each of the objective functions individually subject to the constraints, that is, by solving

$$\text{Min } f_i(x)$$

$$s.t. \ x \in S \ \text{for } i = 1,...,M$$

From the ideal objective vector we obtain the lower bounds of the Pareto optimal set for each objective function.

3. THE PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is an evolutionary computation algorithm motivated by the simulation of social behavior [24-26]. Namely, each agent (individual) utilizes two important kinds of information. The first one is their own experience (Cognitive); that is, they have tried the choices and know which state has been better so far, and they know how good it was. The second one is other agents’ experiences (Social); that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. In the PSO system, each individual makes his decision according to his own experiences (Cognitive) and other agents’ experiences (Social). The system initially has a population of random solutions. Each potential solution, called a particle (agent, individual), is given a random velocity and is flown through the problem space. The agents have memory and each agent keeps track of its previous best position (called the Pbest) and its own experiences (Cognitive); that is, they have tried the choices and know which good it was. The second one is other agent’s experiences (Social); that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. In the PSO system, each individual makes his decision according to his own experiences (Cognitive) and other agents’ experiences (Social). The system initially has a population of random solutions. Each potential solution, called a particle (agent, individual), is given a random velocity and is flown through the problem space. The agents have memory and each agent keeps track of its previous best position (called the Pbest) and its corresponding fitness. There exist a number of Pbest for the respective agents in the swarm and the agent with greatest fitness is called the global best (Gbest) of the swarm. Each particle is treated as a point in a n-dimensional space. The i-th particle is represented as $x_i = [x_{i1}, x_{i2}, ..., x_{in}]$. The best previous position of the i-th particle (Pbest) that gives the best fitness value is represented as $P_i = [p_{i1}, p_{i2}, ..., p_{in}]$. The velocity, i.e., the rate of the position change for particle i is represented as $v_i = [v_{i1}, v_{i2}, ..., v_{in}]$.

The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$v_{ik+1} = w \times v_{ik} + c_1 \times r_1 \times (p_{ik} - x_{ik}) + c_2 \times r_2 \times (P_g - x_{ik})$$

$$x_{ik+1} = x_{ik} + v_{ik+1}$$
Where $i = 1, 2, \ldots, N$, and $N$ is the size of the population; $w$ is the inertia weight; $c_1$ and $c_2$ are two positive constants, called the cognitive and social parameter respectively; $r_1$ and $r_2$ are random numbers uniformly distributed within the range $[0,1]$. Equation (3) is used to determine the $i$-th particle's new velocity $v_{i}^{k+1}$, at each iteration, while equation (4) provides the new position of the $i$-th particle $x_{i}^{k+1}$, adding its new velocity $v_{i}^{k+1}$, to its current position $x_{i}^{k}$. Figure 1 shows the description of velocity and position updates of a particle for a two-dimensional parameter space, also the pseudo code of the general PSO algorithm is shown in figure 2.

Randomly initialize positions and velocities of all particles.
While termination criteria has not satisfied Do{
Set Pbest and Gbest.
Calculate particle velocity according to equation (3).
Update particle position according to equation (4).
Evaluate the objective function value (fitness value).
} satisfactory solution has been found.

Fig. 2. The pseudo code of the general PSO algorithm.

4. Parameters Adaptation

The swarm population size is often between 10 to 40. The reason for a lower population size is that it significantly lowers the computing time. This is because during initialization, all the particles must be in the feasible space. Randomly initialized particles are not always in the feasible space. So initialization may take a longer time if the population is too large. However, for complex cases, a larger population size is preferred. In PSO, there are not many parameters that need to be tuned. Only the following several parameters need to be cared of: maximum velocity $V_{\text{max}}$, inertia weight $w$, acceleration coefficient $C_1$ and $C_2$. Previously, Shi and Eberhart [27] introduced constant inertia weight and linear inertia weight varying usually between 0.8 and 0.4 where it in the first iteration 0.8 and decreasing during the process of run to be 0.4 in the last iteration. On the other hand, Kennedy [28] asserted that the sum of the cognitive and social values $c_1$ and $c_2$ should approximately equal 4.0. For constriction, Carlisle and Dozier [29] have shown that it is advantageous to adjust the cognitive/social ratio to favor cognitive learning (an individualistic swarm). They report that values of 2.8 and 1.3 respectively for the cognitive and social components yield the best performance for the test set they consider.

4.1. Inertia Weight Parameter

In this paper we present an improved PSO algorithm, which uses the dynamic inertia weight that changes according to iterative generation account. We introduce a new modified inertia weight parameter such that:

$$w = 0.5 \left(1 - \sin \left(\frac{\pi t}{4}\right)\right), \quad t=1,2,\ldots,N$$

Where $t$ is the generation number, the modified inertia parameter and be visualized in figure 3.

Fig. 3. Modified inertia weight parameter

4.2. Cognitive and Social Parameter

In this section, we present a new procedure to generate cognitive and social parameter. The idea of this technique is as follows

$$C_1 = 1.3 \gamma + 1.8(1 - \gamma),$$
$$C_2 = 1.3(1 - \gamma) + 1.8 \gamma$$

Where $\gamma = 1.4\delta - 0.2$ and $\delta \in [0,1]$ is a random generated number. Figure 4 gives schematic view of possible sampling region.
This procedure produces a chaos pattern as shown in figure 5. The generating pattern for 50 iteration is shown in figure 5.

5. THE PROPOSED OPTIMIZATION SYSTEM

In this section, we describe a proposed approach for solving multiobjective optimization problems. A population of particles was generated randomly independent of each other and distributed uniformly, which navigates through the search space. PSO is enriched through chaotic constriction factor, this enrichment, accelerate the convergence property of the proposed optimization tool and retain the feasibility of the particles, where it controls the movement velocity of each particle so as to improve search engine visibility. Then chaotic local search is employed as a neighborhood search engine to explore the less crowded area on the Pareto front. The description diagram of the proposed algorithm is shown in figure 6 and it is described as follows:

Phase I: PSO

Step 1. Initialization: A population of particles with random positions and velocities on n-dimensions is initialized in the problem space.

Step 2. Evaluation: The desired optimization fitness function \[ f_i (\overline{x}) \], \( i = 1, 2, ..., M \) in n variables is evaluated for each particle.


Step 4. Updating the velocity and position:

- Update the velocity of each particle according to equation (3).

To enrich the searching behavior and to avoid being trapped into infeasible region, chaotic dynamics (Chaotic constriction factor \( \chi \)) is incorporated into the PSO:

- Update the position of each particle according to

\[
X_i^{k+1} = X_i^k + \chi V_i^{k+1}
\]

To restrict velocity and control it during Evolution of particles, some authors \[30-31\] use a constant/dynamic constriction factor \( \chi \), which has a constant value to improve the performance of PSO. A well-known logistic equation is employed, where it exhibits chaotic dynamics.

\[ X_{n+1} = \mu \cdot X_n (1 - X_n), \quad X_0 = 10^{-6}, \mu = 4, \quad n = 0, 1, 2, \ldots \]

Where, \( n \) is the age of the infeasible particle (How long it’s still unfeasible?)

the new position \( X_i^{k+1} \) depends on velocity \( V_i^{k+1} \).

\[
X_i^{k+1} = X_i^k + V_i^{k+1}
\]

Then, \( V_i^{k+1} \) makes the particle to lose its feasibility, so we introduce a new modified factor \( \chi \) such that new modified position of the particle is computed as:

\[
X_i^{k+1} = X_i^k + \chi V_i^{k+1}
\]

Interested readers could refer to Liu \[31\] for more details. Pseudo code of the proposed chaotic constriction factor is shown in figure 6.

Procedure make

\[
POPULATION = \{ p_i = \{ x_i, v_i \} : (i = 1, 2, ..., pop_{size}) \}
\]

Begin

\[ i \leftarrow 1 \]

While \((i < \text{pop-size})\) do

\[ \chi_0 = 10^{-6} \]

While \( p_i = \{ x_i, v_i \} \) unfeasible

\[ x_i^{k+1} = x_i^k + \chi v_i^{k+1} \]

Check feasibility

\[ X_{n+1} = \mu \cdot X_n (1 - X_n) \]

End

End

Fig 6. Pseudo code of Chaotic constriction factor

Step 5. Evaluation: Evaluate the desired optimization fitness function in n variables for each particle.

Step 6. Updating Pbest and Gbest: For each particle, compare its current objective value with its Pbest value. If the current value is better, then update Pbest with the current position and objective value. Determine the
best particle of the entire current population with the best objective value. If the objective value is better than that of Gbest, then update Gbest with the current best particle.

Step 8. Ranking: Ranks individuals (particles) according to their objective value, and returns a column vector containing the corresponding individual fitness value.

Step 9. Archive $A^{(t)}$ Update: In order to ensure convergence to the true Pareto solutions, we concentrated on how elitism could be presented in the algorithm. So, we propose an archiving-selection strategy (figure 7) that guarantees at the same time progress towards the Pareto-optimal set and a covering of the whole range of the non-dominated solutions. This can be done using update function where, it gets the new population $P^{(t)}$ and the old archive set $A^{(t-1)}$ and determines the updated one, namely $A^{(t)}$.

Input: Current archive $A$, new solution $x$

If $\exists x' \in A \mid x' \succ x$ Do
$A' \leftarrow A$
Else
$D = \{x' \in A \mid x \succ x'\}$
$A' = A \cup \{x\} \setminus D$
End

Output: $A'$

Fig.7: Pseudo code of the Archive updating

**Phase 2: Chaotic Local search**

In this section, Chaotic Local search is described, depending on chaotic equation a new chaotic local search has been driven as follows:

A well-known logistic equation is employed for generating neighborhood solution.

\[ \chi_{n+1} = \mu \cdot \chi_n (1 - \chi_n), \quad \chi_0 = 10^{-6}, \quad \mu = 4, \quad n = 0, 1, 2, \ldots; \]

Although the above equation is deterministic, it exhibits chaotic dynamics. Interested readers could refer to Liu [31] for more details.

The general procedure can be described by the following steps:

Step 1: Start with each population point $\bar{x}_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, called the starting point, and the prescribed step lengths $\Delta x_i$ in each of the coordinate directions $u_i, i = 1, 2, \ldots, n$.

Step 2: Compute $f_k = f(X_k)$. Set $i = 1$, $Y_{k,0} = X_k$, and start the exploratory move as stated in step 3.

Step 3: The variable $x_i$ is perturbed about the current temporary starting point $Y_{k,i-1}$ to obtain the new temporary base point as

\[ Y_{k,i} = Y_{k,i-1} + \chi_i \Delta x_i, \]

\[ \chi_i = \mu \cdot \chi_{i-1}(1 - \chi_i), \quad \chi_{i-1} = 10^{-6}, \quad \mu = 4, \quad n = 0, 1, 2, \ldots; \]

This process of finding the new temporary point is continued for $i = 1, 2, \ldots, \text{until } n$ is perturbed to find $Y_{k,n}$.

The algorithm maintains a finite-sized archive of non-dominated solutions which gets iteratively updated in the presence of new solutions based on the concept of dominance, such that new solutions are only accepted in the archive if they are not dominated by any other element in the current archive.

**6. COMPUTATIONAL EXPERIMENT**

The performance of the proposed algorithm for global optimization continuous function is tested on several constrained well-known engineering benchmark problems [32-38]. The algorithm is coded in MATLAB 8.0 and the simulations are run on CPU with a 2.7GHz Core 2 Duo Intel Processor. Table 1
list the parameter setting used in this simulation

<table>
<thead>
<tr>
<th>Population size</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cognitive parameter</td>
<td>Dynamic Cognitive parameter</td>
</tr>
<tr>
<td>Social parameter</td>
<td>Dynamic Social parameter</td>
</tr>
<tr>
<td>Maximum velocity of particles</td>
<td>Vmax=Xmax-Xmin</td>
</tr>
<tr>
<td>Inertia weight</td>
<td>Dynamic inertia weight</td>
</tr>
<tr>
<td>Initial constriction factor</td>
<td>$\chi_0$</td>
</tr>
</tbody>
</table>

Table 1. The algorithm parameters

6.1. Applications

The proposed approach is applied to some engineering problem was chosen from the engineering application.

- A welded beam design

A welded beam design is used by Deb [33], where a beam needs to be welded on another beam and must carry a certain load $F$ (Fig. 9).

![Fig. 9: The welded beam design problem](image)

It is desired to find four design parameters (thickness $b$, width $t$, length of weld $l$, and weld thickness $h$) for which the cost function of the beam and the deflection function at the open end are minimum. The overhang portion of the beam has a length of “14 inch” and “$F=6000$ lb” force is applied at the end of the beam. A little thought will reveal that a design for minimum deflection at the end (or maximum rigidity of the above beam) will make all four design dimensions to take large dimensions. Thus, the design solutions for minimum cost and maximum rigidity (or minimum-end-deflection) are conflicting to each other. In the following, the mathematical formulation of the two-objective optimization problem is presented as follows:

$$\begin{align*}
\text{Min } f_1(x) &= 1.1047l^2h^2 + 0.0481lb(14 + l) \\
\text{Min } f_2(x) &= 2.1952/t^2b
\end{align*}$$

subject to:

$$\begin{align*}
g_1(x) &= 13600 - r(x) \geq 0, \\
g_2(x) &= 30000 - \sigma(x) \geq 0 \\
g_3(x) &= b - h \geq 0, \\
g_4(x) &= P_c(x) - 6000 \geq 0
\end{align*}$$

$h,b \in [0.125,5]$, $l,t \in [0.1,10]$

where

$$\begin{align*}
\tau &= \sqrt{(\tau')^2 + (\tau'')^2 + l \tau' \tau''/\sqrt{0.25(l^2 + (h + l)^2)}} - \tau' = 6000\sqrt{2hl} \\
\tau'' &= \frac{6000(14 + 0.5t) \sqrt{0.25(l^2 + (h + l)^2)}}{2 \sqrt{2hl(l^2/12 + 0.25(h + l)^2)}} \sqrt{\sigma} = 504000/t^2b \\
P_c &= 64746.022(1 - 0.0282346\gamma) b^{0.9}
\end{align*}$$

In the Welded Beam design problem, the non-linear constraints can cause difficulties in finding the Pareto front. As shown in Fig. 10.

![Fig. 10: Pareto optimal front of welded beam using our approach](image)

- Two-Bar Truss

Figure 11 illustrates the two-bar truss that is to be optimized [34]. This problem was adapted from Kirsch [35]. It is comprised of two stationary pinned joints, A and B, where each one is connected to one of the two bars in the truss. The two bars are pinned where the join one another at joint C, and a 100 kN force acts directly downward at that point. The cross-sectional areas of the two bars are represented as $\chi_1$ and $\chi_2$, the cross-sectional areas of trusses AC and BC respectively. Finally, $y$ represents the perpendicular distance from the line AB that contains the two-pinned base joints to the connection of the bars where the force acts (joint C). The two-bar truss is shown below.

![Two-Bar Truss Diagram](image)
The problem has been modified into a two-objective problem in order to show the non-inferior Pareto set clearly in two dimensions. The stresses in AC and BC should not exceed “100,000 kPa” and the total volume of material should not exceed 0.1 m³. The reason the objective constraints have been imposed is that the Pareto set is asymptotic and extends from -∞ to ∞. As $x_1$ and $x_2$ go to zero, $f_{volume}$ goes to zero and $f_{stress,AC}$ and $f_{stress,BC}$ go to infinity. As $x_1$ and $x_2$ go to infinity, $f_{volume}$ goes to infinity and $f_{stress,AC}$ and $f_{stress,BC}$ go to zero. Hence, in order to generate Pareto optimal solutions in a reasonable range, objective constraints are imposed. The problem formulation is shown below.

$$Minimize f_{volume} = x_1 (16 + y^2)^{0.5} + x_2 (1 + y^2)^{0.5}$$
$$Minimize f_{stress,AC} = \frac{20(16 + y^2)^{0.5}}{1}$$

subject to

$$f_{volume} \leq 0.1$$
$$f_{stress,AC} \leq 100000$$
$$f_{stress,BC} = \frac{80(1 + y^2)^{0.5}}{y x_2} \leq 100000$$
$$1 \leq y \leq 3$$
$$x_1, x_2 > 0$$

Figure 12 declares the Pareto optimal solution of the Two-Bar Truss. Obviously from the results, the proposed algorithm is able to maintain an almost uniform set of non-dominated solution points along the true Pareto-optimal front.

- Speed Reducer Design

The well-known Speed Reducer test Problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed (Fig.13).

The objective is to minimize the speed reducer weight while satisfying a number of constraints imposed by gear and shaft design practices. This problem was modeled by Golinski[36] as a single-level optimization, and since then many others have used it to test a variety of methods. Here, the problem has been converted into a two objective optimization problem. The mathematical formulation, of the problem is now described. There are seven design variables, $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, which represent as depicted in table 2.

| $x_1$ | width of the gear face, cm |
| $x_2$ | teeth module, cm |
| $x_3$ | number of pinion teeth (Integer) |
| $x_4$ | shaft 1 length between bearings, cm |
| $x_5$ | shaft 2 length between bearings, cm |
| $x_6$ | diameter of shaft 1, cm |
| $x_7$ | diameter of shaft 2, cm |

Table 2. Design variables

The first objective $f_1(\cdot)$ is to find the minimum of a gear box volume, (and, hence, its minimum weight). The second objective, $f_2(\cdot)$, is to minimize the stress in one of the two gear shafts. The design is subject to constraints imposed by gear and shaft design practices. An upper and lower limit is imposed on each of the seven design variables. There are 11 other inequality constraints as depicted in table 3.
Upper bound on the bending stress of the gear tooth

Upper bound on the contact stress of the gear tooth

Upper bounds on the transverse deflection of shafts 1, 2
dimensional restrictions based on space and experience
design requirements on the shafts based on experience
Constraints on stress in the gear shafts

Table 3: The problem constraints

The optimization formulation is

\[ \min f_1 = f_{\text{volume}} = 0.7854x_1x_2^2(10x_2^2/3 + 14.933x_2 - 43.0934) - 1.508x_1(x_2^2 + x_1^2) + 7.477(x_1^2 + x_1^2) + 0.7854(x_2x_2^2 + x_2x_1^2), \]

\[ \min f_2 = f_{\text{stress}} = \sqrt{\frac{745x_1}{x_1x_2^3} + 1.69 \times 10^7}{0.1x_3^3} \]

As shown in Figure 14, our proposed approach works well in both distribution and spread. Also, it keep track of all the feasible solutions found by iteratively update the archive content during the optimization.

6.2. Performance Assessment

Our proposed algorithm was tested on set of engineering applications in engineering. The test suite is a collection of different characteristics of Pareto front. Each of the test problems was run twenty times independently, with different seeds, ID metric is used to measure the performance of the algorithm.

Performance Metric (ID): Let \( P^* \) be an ideal solution (in the objective space). Let \( A \) be an approximate set to the PF, the average distance from \( P^* \) to \( A \) is defined as:

\[ \text{ID}(A, P^*) = \frac{\sum d(P^*, A)}{|A|} \]

Where \( d(P^*, A) \) is the minimum Euclidean distance between \( P^* \) and the points in A.

Table 4 shows the ideal solution for each problem (Calculated from the Pareto front for each application).

Table 4. Ideal solution

<table>
<thead>
<tr>
<th>Problem</th>
<th>Ideal solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>welded beam</td>
<td>(2.78001, 0.004407)</td>
</tr>
<tr>
<td>Two-Bar Truss</td>
<td>(0.0040169, 1.72146e003)</td>
</tr>
<tr>
<td>Speed Reducer</td>
<td>(3.17017e003, 0.6930265e003)</td>
</tr>
</tbody>
</table>

Table 5 shows the comparison between the proposed approach and five of the most recent evolutionary algorithms [39-42] using ID performance metric. In this paper, multiobjective PSO is enhanced with chaotic local search scheme. The performance of our approach was evaluated on four test benchmark functions from engineering domain. In Tables 5, the best and worst obtained IGD values for all the test problems are presented with their mean and standard deviation. Also, for each problem, we can rank the different methods according to the ID values and get table 6 and table 7. It is obvious that the algorithm performances well on most of the test problems. As is evident from table 5, table 6, table 7, in all problems, global convergence is obtained and the complete Pareto optimal frontier is discovered. The primary cause of this behavior is local search strategy which enables the algorithm to search less crowded area in the search space. Hence an effective integration of chaotic local search and PSO algorithm is the reason for a better performance of the hybrid algorithm. Overall, the proposed hybrid algorithm performs well on the test problems used for this study. The inclusion of chaotic local search speeds-up the search process and also helps in obtaining a fine-grained value for the objective functions.

![Fig. 14. Result for the speed reducer design](image-url)
Table 5. The mean, standard deviation, the smallest and the largest values of the IGD used for each test problem.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean(ID)</td>
<td>Smallest (ID)</td>
<td>Largest (ID)</td>
<td>Mean(ID)</td>
<td>Mean(ID)</td>
<td>Mean(ID)</td>
</tr>
<tr>
<td>Two-Bar Truss</td>
<td>1.9102e4</td>
<td>1.8233e4</td>
<td>1.9821e4</td>
<td>1.9786e4</td>
<td>1.9203e4</td>
</tr>
<tr>
<td>Speed Reducer Design</td>
<td>2.7156e2</td>
<td>2.5123e2</td>
<td>2.8130e2</td>
<td>2.7801e2</td>
<td>2.7542e2</td>
</tr>
</tbody>
</table>

Table 6. Ranking of the IGD values.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rank 1</th>
<th>Rank 2</th>
<th>Rank 3</th>
<th>Rank 4</th>
<th>Rank 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>The proposed algorithm</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hybrid particle swarm [41]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Ant Optimization System [43]</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>quantum genetic algorithm [42]</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rough Sets Based Approach [44]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7. Statistics of Ranking.

6. CONCLUSION

In this paper, a hybrid multiobjective algorithm combining adaptive PSO and chaotic local search is proposed. To enrich the searching behaviour and to avoid being trapped into local optimum, dynamic adaptive of PSO parameters are incorporated into PSO. In the proposed algorithm, inertia weight of each particle is dynamically updated, also personal influence and social influence parameters are dynamically adapted during the process. Moreover, a modified velocity updating formula of the particle is presented, where, a new constriction factor which control the feasibility of the particles is presented. Chaotic search CS is employed for the local search to explore the less crowded area in the Pareto front. The results, provided by the proposed algorithm for benchmark engineering problems, are promising when compared with exiting well-known algorithms.

REFERENCES


James N. Richardson, Guy Nordenson, Rebecca Laberenne, Rajan Filomeno Coelho, Sigrid Adriaenssens, Flexible optimum design of a bracing system for facade design using multiobjective Genetic Algorithms, Automation in Construction, Volume 32, July 2013, Pages 80-87.


Kun Fan, Wei Jia You, Yuanyuan Li, An effective modified binary particle swarm optimization (mBPSO) algorithm for multiobjective resource allocation problem (MORAP), Applied Mathematics and Computation, Volume 221, 15 September 2013, Pages 257-267.


