Hybrid optimization technique coupling an evolutionary algorithm and chaotic local search

H. AL-Oraby, Kotb A. Kotb, A. A. Mousa

Abstract: Evolutionary algorithms (EAs) are randomized heuristics for search and optimization that are based on principles derived from natural evolution. Mutation, recombination, and selection are iterated with the goal of driving a population of candidate solutions toward better and better regions of the search space. Since the underlying idea is easy to grasp and almost no information about the problem to be optimized is necessary in order to apply it, EAs are widely used in many practical disciplines, mainly in computer science and engineering. However, they are time consuming algorithms that are unpractical from the industrial viewpoint, and they are very poor in terms of Global convergence performance. On the other hand, local search algorithms can converge quickly to these local minima and get stuck in a local optimum solution. In this paper, chaotic local search is proposed as a neighborhood search engine to improve the solution quality, where it intends to explore the less-crowded area in the current archive to possibly obtain more nondominated solutions. Finally, various kinds of multiobjective (MO) benchmark problems have been reported to stress the importance of hybridization algorithms in generating Pareto optimal sets for multiobjective optimization problems.

Keywords: Hybrid optimization technique, evolutionary algorithm, chaotic local search.

1. INTRODUCTION

Multiobjective optimization (MO) is an important research topic for both scientists and engineers [1]. In MO, a set of nondominated solutions is usually produced instead of a single recommended solution. According to the concept of non-dominance, a solution to an MO problem is nondominated, or Pareto optimal, if no objective can be improved without worsening at least one other objective[2,3]. Traditional MO methods attempt to find the set of nondominated solutions using mathematical programming. In the case of nonlinear problems, the weighting method and the ε-constraint method are the most commonly used techniques [1]. Both methods transform the MO problem into a single objective problem (SOP) which can be solved using nonlinear optimization. With the weighting method, nondominated solutions are obtained if all weights are positive but not all Pareto optimal solutions can be found unless all objective functions as well as the feasible region are convex. Another disadvantage of this method is that many different sets of weights may produce the same solution, compromising the efficiency of the method [1].

During the past decade, various multiobjective evolutionary algorithms (MOEAs) have been proposed and applied in MOPs [4-9]. A representative collection of these algorithms includes the vector evaluated genetic algorithm (VEGA) by Schaffer [10], the niched Pareto genetic algorithm NPGA [11] and the non-dominated sorting genetic algorithm NSGA by Srinivas and Deb[12], the non-dominated sorting genetic algorithm NSGA-II by Deb et al.[13], the strength Pareto evolutionary algorithm (SPEA) by Zitzler and Thiele [14], the strength Pareto evolutionary algorithm (SPEA-II) by Zitzler et al. [15], the Pareto archived evolution strategy (PAES) by Knowles and Corne [16] and the memetic PAES (M-PAES) [17]. Although these MOEAs differ from each other in both exploitation and exploration, they share the common purpose, searching for a near-optimal, well-extended and uniformly diversified Pareto optimal front for a given MOP. However, this ultimate goal is far from being accomplished by the existing MOEAs as documented in the literature, e.g., [4].

In this paper, hybrid evolutionary algorithm is presented, where chaotic local search is implemented as a neighborhood search engine to improve the solution quality. Chaotic local search intends to explore the less-crowded area in the current archive to possibly obtain more nondominated solutions. Finally, various kinds of multiobjective (MO) benchmark problems have been reported to stress the importance of hybridization algorithms in generating Pareto optimal sets for multiobjective optimization problems. The organization of the rest of this paper is as follows. In section 2, Multiobjective optimization is presented. In section 3, we propose a hybrid multiobjective optimization algorithm. Experimental Results is presented and analyzed in section 4. Finally, section 5 concludes the paper.
2. MULTIOBJECTIVE OPTIMIZATION (MO)

A general minimization problem of $M$ objectives can be mathematically stated as:

$$\begin{align*}
\text{Minimize :} & \quad \mathbf{f}(\mathbf{x}) = \left[ f_i(\mathbf{x}), \quad i = 1,2,\ldots,M \right] \\
\text{subject to the constraints :} & \quad g_j(\mathbf{x}) \leq 0, \quad j = 1,2,\ldots,J.
\end{align*}$$

given $\mathbf{x} = x_1, x_2, \ldots, x_n$, where $n$ represents the dimension of the decision variable space, $f_i$ is the $i$-th objective function, $g_j$ is the $j$-th inequality constraint. The MO problem then reduces to finding an $\mathbf{x}$ such that $f_i(\mathbf{x})$ 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 $\mathbf{u} = u_1, u_2, \ldots, u_M$ is said to dominate a vector $\mathbf{v} = v_1, v_2, \ldots, v_M$ ($\mathbf{u}$ dominate $\mathbf{v}$ denoted by $\mathbf{u} \geq \mathbf{v}$), for a MO minimization problem, if and only if

$$\forall i \in 1, \ldots, M, \quad u_i \leq v_i \wedge \exists i \in i, \ldots, M : u_i < v_i$$

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

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

3. GENETIC ALGORITHMS

Genetic Algorithms are a part of the evolutionary algorithms, which is a rapidly growing areas of artificial intelligence [18]. They are inspired by Darwin's theory of biological evolution. By mimicking this process, GAs are able to evolve solutions to real world problems. Holland [19] was the first one to put computational evolution on a firm theoretical footing. GAs are one of the optimization techniques based on the concepts of biological evolution and genetics. In this algorithm, the variables are represented as genes on a chromosome. It features a group of candidate solutions (i.e., population) on the response surface. Through natural selection and the genetic operators, mutation and recombination, chromosomes with better fitness are found. Natural selection guarantees that chromosomes with better fitness will propagate in the future populations. Using the recombination operator, GA combines genes from two parents to form children (new chromosomes) that have a high probability of having better fitness than their parents. Mutation allows new research areas of the response surface to be explored. GAs offer a generation improvement in the fitness of the chromosomes and after many generations will create chromosomes containing the optimized settings [20-22]. Genetic algorithm was invented by "John Holland" in the 1960s and it was later developed by Holland and his students at the University of Michigan in 1960s and 1970s. Holland's 1975 book "Adaptation in Natural and Artificial Systems" [19] presented the genetic algorithms as an abstraction of biological evolution and gave a theoretical framework for adaptation under the genetic algorithms[24]. Figure 1 shows a flowchart of the working of a GA.

![Fig. 1 Main Flowchart of SGA.](image-url)
4. HYBRID MULTIOBJECTIVE OPTIMIZATION ALGORITHM

The proposed methodology introduces a hybrid algorithm combining GAs and chaotic local search to improve the performance of each algorithm. To improve the solution quality chaotic local search scheme is implemented as a neighborhood search engine, where it intends to explore the less-crowded area in the current archive to possibly obtain more nondominated solutions nearby. The description diagram of the proposed algorithm is described as follows:

4.1. Initialization stage

The proposed approach uses two separate populations, the first population \( P^{(t)} \) consists of the individuals which initialized randomly satisfying the search space (i.e., the lower bounds and upper bounds), while the second population \( R^{(t)} \) consists of reference points which satisfying all constraints (feasible points).

4.2. Repair algorithm

The aim of this technique is to separate any feasible individuals in a population from those that are infeasible by evolving (repairing) infeasible individuals. This approach co-evolves the population of infeasible individuals until they become feasible. Repair process works as follows. Assume, there is a search point \( \omega \not\in S \) (where \( S \) is the feasible space which determined by the lower bounds and upper bounds). In such a case the algorithm selects one of the reference points, says \( r \in S \) and creates random points \( Z \) from the segment defined between \( \omega r \), but the segment may be extended equally on both sides determined by a user specified parameter \( \mu \in [0,1] \). Thus, a new feasible individual is expressed as follows:

\[
\begin{align*}
z_1 &= \gamma \omega + (1-\gamma) \cdot r, \\
z_2 &= (1-\gamma) \cdot \omega + \gamma \cdot r
\end{align*}
\]

Where \( \gamma = (1+2\mu)\delta - \mu \) and \( \delta \in [0,1] \) is a random generated number. The interested reader can refer to [25, 26].

4.3. Evaluation stage

A population of size \( N \) can be evaluated according to nondomination concept. Consider a set of population members, having \( K (K>1) \) objective functions values. The following algorithm explains the algorithm by which the nondominated set of solutions can be found [3]. The algorithm initially locates an externally finite size archive of observed nondominated solutions “approximated nondominated solutions”, see figure 2.

<table>
<thead>
<tr>
<th>Step 0: Begin with ( i = 1 ).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>: For all ( j = 1, 2, ..., N ) and ( j \neq i ), compare solutions ( x^j ) and ( x^i ) for domination.</td>
</tr>
<tr>
<td><strong>Step 2</strong>: If for any ( j ), ( x^j ) is dominated by ( x^i ), mark ( x^j ) as &quot;dominated&quot;.</td>
</tr>
<tr>
<td><strong>Step 3</strong>: If all solutions (that is, when ( i = N ) is reached) in the set are considered, Go to Step 4, else increment ( i ) by one and Go to Step 1.</td>
</tr>
<tr>
<td><strong>Step 4</strong>: All solutions that are not marked &quot;dominated&quot; are non-dominated solutions.</td>
</tr>
</tbody>
</table>

4.4 Archiving/selection strategy

In order to ensure convergence to the true Pareto-optimal solutions, we concentrated on how elitism could be introduced in the proposed algorithm. So, we propose an archiving/selection strategy that guarantees at the same time progress towards the Pareto-optimal set and a covering of the whole range of the non-dominated solutions.

4.5. Selection Stage

Reproduction (Selection) operator is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to get copied into the next generation. The principle behind GAs is essentially Darwinian natural selection. The selection directs GAs search towards promising regions in the search space, a random-weight approach is adopted to obtaining a variable search direction towards the Pareto frontier. The weighted-sum objective is given as follows:

\[
f(x) = w_1 f_1(x) + \ldots + w_k f_k(x) = \sum_{i=1}^{k} w_i f_i(x)
\]

where \( x \) is a string, \( f(x) \) is a combined fitness function, \( f_i(x) \) is the \( i-th \) objective function and...
\( \left\{ w_i \mid \sum_{i=0}^{k} w_i = 1 \right\} \) is a constant weight for \( f_i(x) \).

Roulette wheel selection is employed as selection mechanism. Where, the individuals on each generation are selected for survival into the next generation according to a probability value proportional to the ratio of individual fitness over total population fitness. The probability of variable selection is proportional to its fitness value in the population, according to the formula given by

\[
p(x) = \frac{f(x) - f_{\text{Min}}(\psi)}{\sum_{x \in \psi} [f(x) - f_{\text{Min}}(\psi)]},
\]

where \( p(x) \), selection probability of a string \( x \) in a population \( \psi \) and \( f_{\text{Min}}(\psi) = \text{Min}\{f(x) \mid x \in \psi\}\)

### 4.6. Crossover Operators

The aim of crossover is to exchange information between two parents chromosomes in order to produce two new offspring for the next population. There are many ways in which crossover can be implemented, such as one point crossover, two-point crossover, n-point crossover, or uniform crossover. BLX-\( \alpha \) crossover is implemented to produce the next generation.

### 3.7. Mutation Operators

Mutation occasionally injects a random alteration for one of the genes. Similar to mutation in nature, this function preserves diversity in the population. It provides innovation, possibly leading to exploration of a region of better solutions. Mutation is performed with low probability \( p_m \). Applied in conjunction with selection and crossover, mutation not only leads to an efficient search of the solution space but also provides an insurance against loss of needed diversity.

### 4.8. 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 (fig.3) 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.3:** Pseudo code of the Archive updating

### 4.9. Chaotic Local search stage

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 \( \chi_d \) employed for generating neighborhood solution.

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

Although the above equation is deterministic, it exhibits chaotic dynamics when control parameter \( \chi_0 \in \{ 0, 0.25, 0.5, 0.75, 1 \} \). Interested readers could refer to Liu [27] for more details.

The general procedure can be described by the following steps:
Step 1: Start with each population point $X = (x_1, x_2, \ldots, x_n)$, called the starting point, and the prescribed step lengths $\Delta x_i$ in each of the coordinate directions $u_i, i = 1, 2, \ldots, n$. Set $k = 1$.

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 $i = 1, 2, \ldots$ 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.

5. Experimental Results

In order to validate the proposed algorithm versus another advanced MOEAs, graphical presentation of the experimental results and associated observations are presented in this section. Table 1 lists the parameter setting used in the algorithm for all runs. In our comparison study, two prominent benchmark test functions with distinct Pareto optimal front are selected from [25, 28, 29], which brought forward various benchmark functions for testing MOEAs. Also, the problems chosen from the engineering domains is the Speed Reducer design used by Golinski [30].

<table>
<thead>
<tr>
<th>Population size (N)</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Generation</td>
<td>200</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.99</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.02</td>
</tr>
<tr>
<td>Selection operator</td>
<td>Roulette Wheel</td>
</tr>
<tr>
<td>Crossover operator</td>
<td>BLX-α</td>
</tr>
<tr>
<td>Mutation operator</td>
<td>Polynomial mutation</td>
</tr>
</tbody>
</table>

**Table 1: GA parameters**

Problem 1:

$$\begin{align*}
\text{Min } f_1(x) &= -x_1 - x_2^2, \\
\text{Max } f_2(x) &= -x_1^2 - x_2 \\
\text{subject to } \quad g_1(x) &= 12 - x_1 - x_2 \geq 0, \\
& \quad g_2(x) = x_1^2 + 10x_1 - x_2^2 - 80 \geq 0, \\
\text{With } x_1 &\in [2, 7] \\
& \quad x_2 \in [5, 10]
\end{align*}$$
Problem 2:

\[
\begin{align*}
\text{Min } f_1(x) &= 2 + (x_1 - 2)^2 + (x_2 - 2)^2 \\
\text{Min } f_2(x) &= 9x_1 - (x_2 - 1)^2 \\
\text{Subject to:} \\
g_1(x) &= x_1^2 + x_2^2 \leq 225 \\
g_2(x) &= x_1 - 3x_2 + 10 \leq 0 \\
x_1 &\in [-20, 20] \\
x_2 &\in [-20, 20]
\end{align*}
\]

Problem 3:

\[
\begin{align*}
\text{Min } f_1(x) &= -x_1^2 + x_2 \\
\text{Min } f_2(x) &= 0.5x_1 + x_2 + 1 \\
\text{subject to:} \\
g_1(x) &= \frac{x_1}{6} + x_2 - 6.5 \leq 0 \\
g_2(x) &= 0.5x_1 + x_2 - 7.5 \leq 0 \\
g_3(x) &= 5x_1 + x_2 - 30 \\
x_1 &\geq 0, \ x_2 \geq 0
\end{align*}
\]
Problem 1 (Fig. 4) and problem 2 (Fig. 5) and problem 3 (Fig. 6) are relatively difficult. The constraints in the problems are nonlinear. We compare our method with a reliable and efficient multiobjective genetic algorithm (NSGA II). The results show that our method can be used efficiently for constrained MOP than (NSGA II). It worth mentioning that the number of the Pareto optimal solutions obtained by (NSGA II) is limited by its population size, but our optimization system keep track of all the feasible solutions found during the optimization and therefore do not have any restrictions on the number of the Pareto optimal solutions found. On the other hand figures (4)-(6) declare that 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.7).

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[30] 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 | \(x_5\) | shaft 2 length between bearings, cm |
| \(x_2\) | teeth module, cm | \(x_6\) | diameter of shaft 1, cm |
| \(x_3\) | number of pinion teeth (Integer) | \(x_7\) | diameter of shaft 2, cm |
| \(x_4\) | shaft 1 length between bearings, cm |

Table 2. Design variables
The first objective, $f_1(x)$, is to find the minimum of a gear box volume, (and, hence, its minimum weight). The second objective, $f_2(x)$, 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.

| $g_1$ | Upper bound on the bending stress of the gear tooth |
| $g_2$ | Upper bound on the contact stress of the gear tooth |
| $g_3, g_4$ | Upper bounds on the transverse deflection of shafts 1, 2 |
| $g_5 - g_7$ | Dimensional restrictions based on space and experience |
| $g_8, g_9$ | Design requirements on the shafts based on experience |
| $g_{10}, g_{11}$ | Constraints on stress in the gear shafts |

Table 3: The problem constraints

The optimization formulation is

$$
\begin{align*}
\text{Min } f_1 &= f_{\text{volume}} = \\
&= 0.7854x_1x_2^2(10x_3^2 / 3 + 14.933x_4 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^2 + x_7^2) + 0.7854(x_4x_6^2 + x_5x_7^2), \\
\text{Min } f_2 &= f_{\text{stress}} = \sqrt{\frac{(745x_4 / x_2x_3)^2 + 1.69 \times 10^7}{0.1x_3^3}},
\end{align*}
$$

s.t.

$$
\begin{align*}
g_1 : \frac{1}{x_1x_2^2x_3} - \frac{1}{27} &\leq 0, \\
g_2 : \frac{1}{x_1x_2^2x_3} - \frac{1}{397.5} &\leq 0, \\
g_3 : \frac{x_2^3}{x_3x_4x_5} - \frac{1}{1.93} &\leq 0, \\
g_4 : \frac{x_2^3}{x_3x_4x_5} - \frac{1}{1.93} &\leq 0, \\
g_5 : x_2x_3 - 40 &\leq 0, \\
g_6 : \frac{x_1}{x_2} - 12 &\leq 0, \\
g_7 : 5 - \frac{x_1}{x_2} &\leq 0, \\
g_8 : 1.9 - x_4 + 1.5x_6 &\leq 0, \\
g_9 : 1.9 - x_4 + 1.1x_7 &\leq 0, \\
g_{10} : f_2(\bar{x}) &\leq 1100, \\
g_{11} : \sqrt{(745x_4 / x_2x_3)^2 + 1.575 \times 10^6} / 0.1x_3^3 &\leq 850.
\end{align*}
$$

2.6 $\leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad 17 \leq x_3 \leq 18, \quad 7.3 \leq x_4 \leq 8.3, \quad 7.3 \leq x_5 \leq 8.3, \quad 2.9 \leq x_6 \leq 3.9, \quad 5.0 \leq x_7 \leq 5.5,$

As shown in Figure 8, 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.

![Fig. 8. Result for the speed reducer design](image_url)
6. CONCLUSIONS

Evolutionary algorithms are widely used in many practical disciplines, mainly in computer science and engineering. However, they are time consuming algorithms that are unpractical from the computational viewpoint, and they are very poor in terms of Global convergence. Hybrid optimization algorithm is presented, where chaotic local search is implemented as a neighborhood search engine to improve the solution quality, in the way such that local search algorithms can converge quickly to local minima and get stuck in a local optimum solution. The proposed algorithm intends to explore the less-crowded area in the current archive to possibly obtain more nondominated solutions. The results, provided by the proposed algorithm for benchmark test problems are promising when compared with the exiting well-known algorithm. Also, our results suggest that the proposed algorithm optimization system is better applicable for solving real-world-application problems. For future work, we intend to test the algorithm on more problems. We would also like to improve our method to be applicable for more complex real-world applications.

REFERENCE


