A genetic algorithm for unconstrained multi-objective optimization

Qiang Long a, Changzhi Wu b,*, Tingwen Huang c, Xiangyu Wang b,d

a School of Science, Southwest University of Science and Technology, Mianyang, China
b Australasian Joint Research Centre for Building Information Modelling, School of Built Environment, Curtin University, Australia
c Texas A&M University at Qatar, Doha, P.O.Box 23874, Qatar
d Department of Housing and Interior Design, Kyung Hee University, Seoul, South Korea

A R T I C L E   I N F O

Article history:
Received 29 August 2014
Received in revised form 18 November 2014
Accepted 19 January 2015
Available online 28 January 2015

Keywords:
Genetic algorithm
Optimal sequence method
Multi-objective optimization
Numerical performance evaluation

A B S T R A C T

In this paper, we propose a genetic algorithm for unconstrained multi-objective optimization. Multi-objective genetic algorithm (MOGA) is a direct method for multi-objective optimization problems. Compared to the traditional multi-objective optimization method whose aim is to find a single Pareto solution, MOGA tends to find a representation of the whole Pareto frontier. During the process of solving multi-objective optimization problems using genetic algorithm, one needs to synthetically consider the fitness, diversity and elitism of solutions. In this paper, more specifically, the optimal sequence method is altered to evaluate the fitness; cell-based density and Pareto-based ranking are combined to achieve diversity; and the elitism of solutions is maintained by greedy selection. To compare the proposed method with others, a numerical performance evaluation system is developed. We test the proposed method by some well known multi-objective benchmarks and compare its results with other MOGASs; the result show that the proposed method is robust and efficient.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

In this paper, we consider the following multi-objective optimization problem:

\[
\begin{align*}
\text{(MOP)} \\
\text{Minimize } & F(x) \\
\text{Subject to } & x \in X,
\end{align*}
\]

where \( F(x) = (f_1(x), f_2(x), \ldots, f_p(x))^T \) is a vector-valued function, \( X = \{ x \in \mathbb{R}^n : l_b \leq x \leq u_b \} \subset \mathbb{R}^n \) is a box set, \( l_b \) and \( u_b \) are lower and upper bounds, respectively. We suppose that \( f_i(x), i = 1, 2, \ldots, p \) are Lipschitz continuous but not necessarily differentiable.

Multi-objective optimization has extensive applications in engineering and management [2,28,29]. Most of the optimization problems appearing in real-world applications have multiple objectives; they can be modeled as multi-objective optimization problems. However, due to the theoretical and computational challenges, it is not easy to solve multi-objective optimization problems. Therefore, multi-objective optimization attracts lots of researches over the last few decades.

So far, there are two types of methods to solve multi-objective optimization problems: indirect and direct methods. The indirect method converts multiple objectives into a single one. One strategy is to combine the multiple objective functions using the utility theory or the weighted sum method. The difficulties for such methods are the selection of utility function or proper weights so as to satisfy the decision-maker’s preferences, and furthermore, the greatest deficiency of the (linear) weighted sum method is that we cannot obtain the concave part of the Pareto frontier. Another indirect method is to formulate the multiple objectives, except one, as constraints. However, it is not easy to determine the upper bounds of these objectives. On the one hand, small upper bounds could exclude some Pareto solutions; on the other hand, large upper bounds could enlarge the objective function value space which leads to some sub-Pareto solution. Additionally, indirect method can only obtain a single Pareto solution in each run. However, in practical applications, decision-makers often prefer a number of Pareto solutions so that they can choose one strategy according to their preferences.

Direct methods devote themselves to explore the entire set of Pareto solutions or a representative subset. However, it is extremely hard or impossible to obtain the entire set of Pareto solutions for most multi-objective optimization problems, except some simple cases. Therefore, stepping back to a representative subset is preferred. Genetic algorithm (GA), as a population-based algorithm, is a good choice to achieve this goal. The generic single-objective genetic algorithm can be modified to find a set of multiple non-dominated solutions in a single run. The ability of the genetic algorithm to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems. The crossover operator of the genetic
algorithm can exploit structures of good solutions with respect to different objectives, which in turn, creates new non-dominated solutions in unexplored parts of the Pareto frontier. In addition, multi-objective genetic algorithm does not require user to prioritize, scale, or weight objectives. Therefore, the genetic algorithm is one of the most popular metaphorisitic approaches for solving multi-objective optimization problems [18,24,36].

The first multi-objective optimization method based on the genetic algorithm, called the vector evaluated GA (or VEGA), was proposed by Schaffer [35]. Afterwards, several multi-objective evolutionary algorithms were developed, such as Multi-objective Genetic Algorithm (MOGA) [6], Niched Pareto Genetic Algorithm (WGBA) [15], Weight-based Genetic Algorithm (WGBA) [13], Random Weighted Genetic Algorithm (RWGA) [31], Nondominated Sorting Genetic Algorithm (NSGA) [38], Strength Pareto Evolutionary Algorithm (SPEA) [52], improved SPEA (SPEA2) [51], Pareto- Archived Evolution Strategy (PAES) [21], Pareto Envelope-based Selection Algorithm (PESA) [3], Nondominated Sorting Genetic Algorithm- II (NSGA-II) [5], Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [1,46,47] and Indicator-Based Evolutionary Algorithm (IBEA) [34].

There are three basic issues [50] in solving multi-objective optimization problems. In Section 4, an evaluation of multi-objective optimization and the process of genetic algorithm. In Section 5, Section 6 concludes the paper.

2. Preliminaries

In this section, we first review some definitions and theorems in the multi-objective optimization, and then introduce the general procedure of genetic algorithm.

2.1. Definitions in multi-objective optimization

First of all, we present the following notations which are often used in vector optimization. Given two vectors $x = (x_1, x_2, \ldots, x_n)^T$ and $y = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$, then

- $x = y \iff x_i = y_i$ for all $i = 1, 2, \ldots, n$;
- $x < y \iff x_i < y_i$ for all $i = 1, 2, \ldots, n$;
- $x \leq y \iff x_i \leq y_i$ for all $i = 1, 2, \ldots, n$;
- $x < y \iff x_i < y_i$ for all $i = 1, 2, \ldots, n$;
- $x \geq y \iff x_i \geq y_i$ for all $i = 1, 2, \ldots, n$.

“$>$”, “$\geq$” and “$\leq$” can be defined similarly. In this paper, we call $x \leq y$ and $x_i < y_i$ for all $i = 1, 2, \ldots, n$.

Definition 2.1. Suppose that $x \in \mathbb{R}^n$ and $x^* \in \mathbb{X}$. If $x^* \leq x$ for any $x \in \mathbb{X}$, then $x^*$ is called an absolute optimal point of $\mathbb{X}$.

Absolute optimal point is an ideal point but it may not exist.

Definition 2.2. Let $x \in \mathbb{R}^n$ and $x^* \in \mathbb{X}$. If there is no $x \in \mathbb{X}$ such that $x \leq x^*(x < x^*)$, then $x^*$ is called an efficient point (or weakly efficient point).

The sets of absolute optimal points, efficient points and weakly efficient points of $\mathbb{X}$ are denoted as $\mathbb{X}_{aop}$, $\mathbb{X}_{op}$ and $\mathbb{X}_{wop}$, respectively. For the problem MOP, $\mathbb{X} \subseteq \mathbb{R}^n$ is called the decision variable space and its image set $F(\mathbb{X}) = \{y \in \mathbb{R}^m | y = F(x), x \in \mathbb{X} \subseteq \mathbb{R}^n$ is called the objective function value space.

Definition 2.3. Suppose that $x^* \in \mathbb{X}$. If $F(x^*) \leq F(x)$, for any $x \in \mathbb{X}$, $x^*$ is called an absolute optimal solution of the problem MOP. The set of absolute optimal solution is denoted as $\mathbb{X}_{aop}$.

The concept of the absolute optimal solution is a direct extension of that for single-objective optimization. It is the ideal solution but may not exist for most cases.

Definition 2.4. Suppose that $x^* \in \mathbb{X}$. If there is no $x \in \mathbb{X}$ such that $F(x) \leq F(x^*)$ (or $F(x) < F(x^*)$), i.e., $F(x^*)$ is an efficient point (or weakly efficient point) of the objective function value space $F(\mathbb{X})$, then $x^*$ is called an efficient solution (or weakly efficient solution) of the problem MOP. The sets of efficient solutions and weakly efficient solutions are denoted as $\mathbb{X}_{eop}$ and $\mathbb{X}_{weop}$, respectively.

Another name of the efficient solution is Pareto solution, which was introduced by T.C. Koopmans in 1951 [22]. The meaning of Pareto solution is that, if $x^* \in \mathbb{X}_{aop}$, then there is no feasible solution $x \in \mathbb{X}$, such that any $f_j(x)$ of $F(x)$ is not worse than that of $F(x^*)$. In other words, $x^*$ is the best solution in the sense of “$\leq$”. Another intuitive interpretation of Pareto solution is that it cannot be improved with respect to any objective without worsening at least one of the other objectives. The set of Pareto solutions is denoted by $\mathbb{P}$. Its image set $F(\mathbb{P})$ is called the Pareto frontier, denoted by $\mathbb{P}^*$.  

2.2. Genetic algorithm

Genetic algorithm is one of the most important evolutionary algorithms. It was introduced by John Holland in 1960s, and then developed by his students and colleaguees at the University of Michigan between 1960s and 1970s [14]. Over the last two decades, the genetic algorithm was increasingly enriched by plenty of literatures, such as [9,10,12,19]. Nowadays various genetic algorithms are applied in different areas; for example, mathematical programming, combinational optimization, automatic control and image processing.

Suppose that $P(t)$ and $O(t)$ represent parents and offsprings of the $t$th generation, respectively. Then, the general structure of genetic algorithm can be described in the following pseudocode.

General structure of genetic algorithm.
A multi-objective genetic algorithm

In this section, we present a genetic algorithm to solve the problem of assigning weights to parameters. The approach is straightforward and requires generation of a single objective using normalized weights. However, the decision of weight parameters is not an easy task for this approach.

2.1 Crossover operator: generate $O_1(t)$;
2.2 Mutation operator: generate $O_2(t)$;
2.3 Evaluate $O_1(t)$ and $O_2(t)$; compute the fitness function;
2.4 Selection operator: build the next population;
2.5 $t = t + 1$ go to 2.1

end

From the pseudocode, we can see that there are three important operators in a general genetic algorithm: the crossover operator, mutation operator and selection operator. Usually a different encoding leads to different operators.

### 3. Fitness

For single-valued function, fitness is normally assigned as its function value. However, to assign fitness of multi-objective function is not straightforward. So far, there are three typical approaches. The first one is weight sum approach, which converts the multiple objectives into a single objective using normalized weight $\sum_{i=1}^{p} \lambda_i$, $\lambda_i > 0, i = 1, 2, \ldots, p$. The decision of weight parameters is not an easy task for this approach. The second one is altering objective functions, which randomly divides the current population into $p$ equal sub-populations: $P_1, P_2, \ldots, P_p$. Then, each solution in subpopulation $P_i$ is assigned a fitness value based on objective function $f_i$. In fact, this approach is a straightforward extension of the single-objective genetic algorithm. The last one is Pareto-ranking approach, which is a direct application of the definition of Pareto solution. More specifically, the population is ranked according to a dominance rule. Then, each solution is assigned a fitness value based on its rank in the population rather than its actual objective function value. In the following, we present a new rank strategy based on optimal sequence method [25].

**Definition 3.1.** Let $P = \{1, 2, \ldots, p\}$ and $N = \{1, 2, \ldots, n\}$, for any $x^i, x^j \in X$, define

$$
a_{ij} = \begin{cases} 
1, & f_j(x^i) < f_j(x^j); \\
0.5, & f_j(x^i) = f_j(x^j); \\
0, & f_j(x^i) > f_j(x^j) \text{ or } i = j.
\end{cases}
$$

(2)

Then,

$$
a_i = \sum_{j \in P} a_{ij}, \quad i, j \in N
$$

is called the optimal number of $x^i$ corresponding to $x^j$ for all objectives. Furthermore, $K_i = \sum_{j \in X} a_{ij}$ is defined as the total optimal number of $x^i$ corresponding to all the other solutions for all objectives.

### Algorithm 3.1. Optimal Sequence Method (OSM).

Step 1: Input: the current population and their objective function values.

Step 2: Compute optimal numbers: compute all the optimal numbers and total optimal numbers in Table 1 according to Eq. (2).

Step 3: Rank the solution: re-arrange the order of solutions according to the decreasing order of the total optimal numbers $K_i$. More precisely, denote

$$
K_{a_1} = \max_{1 \leq i \leq N} \{K_i\},
$$

and so on. Then, the solutions $x^{a_1}, x^{a_2}, \ldots$ are called the best solution, the second best solution, etc. The new order is called optimal order.

It is worthy to mention that the optimal sequence method does not necessarily rank Pareto solutions in the frontier positions, but rank those solutions which are more reasonable in the foremost positions. This is different from Pareto-ranking approach.

**Lemma 3.1.** Suppose that $x^i, x^j \in X$. If $F(x^i) \leq F(x^j)$, then $a_{ij} > a_{ji}$.

**Proof.** Let

$$
A = \{l \in P \mid f_j(x^i) < f_j(x^j)\}, \quad a = \mid A \mid;
$$

$$
B = \{l \in P \mid f_j(x^i) = f_j(x^j)\}, \quad b = \mid B \mid;
$$

where $\mid \cdot \mid$ denotes the number of components in a set. Obviously, we have $a > 0, b > 0$ and $a + b = p$. Then,

$$
a_{ij} = \begin{cases} 
1, & l \in A \\
0.5, & l \in B
\end{cases}
$$

and

$$
a_{ji} = \begin{cases} 
0, & l \in A \\
0.5, & l \in B
\end{cases}
$$

Therefore,

$$
a_i = \sum_{l \in A} a_{il} + \sum_{l \in B} a_{il} = a + 0.5b
$$

### Table 1

<table>
<thead>
<tr>
<th>$x^i$</th>
<th>$x^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^1$</td>
<td>$a_{12}$</td>
</tr>
<tr>
<td>$x^2$</td>
<td>$a_{21}$</td>
</tr>
<tr>
<td>$x^j$</td>
<td>$a_{jj}$</td>
</tr>
<tr>
<td>$x^n$</td>
<td>$a_{nn}$</td>
</tr>
</tbody>
</table>

Table of optimal numbers.
and \[
\alpha_j = \sum_{l \in A} \alpha_{jl} + \sum_{l \in B} \alpha_{jl} = 0.5b.
\]

Since \(a > 0\), \(a_j > a_j\). □

**Lemma 3.2.** Suppose that \(x^i, x^j \in X\). If \(F(x^i) \leq F(x^j)\), then, for any \(x^k (k \neq i, j, k \in N)\), \(a_{ik} > a_{kj}\).

**Proof.** For any \(x^k (k \neq i, j, k \in N)\), let
\[
A = \{ l \in P \mid f_i(x^k) < f_j(x^k) \}, \quad a = |A|;
\]
\[
B = \{ l \in P \mid f_i(x^k) > f_j(x^k) \}, \quad b = |B|;
\]
\[
C = \{ l \in P \mid f_i(x^k) = f_j(x^k) \}, \quad c = |C|.
\]

Then, we have \(a, b, c \geq 0\) and \(a + b + c = p\).

- When \(l \in A\). Since \(F(x^i) \leq F(x^j)\), we have \(f_i(x^k) < f_j(x^k)\) and
\[
f_i(x^k) < f_j(x^k).
\]

Therefore,
\[
\alpha_{ik} = \sum_{l \in A} \alpha_{jl} = 0 \cdot a = 0,
\]
\[
\alpha_{jk} = \sum_{l \in B} \alpha_{jl} = 0 \cdot a = 0.
\]

- When \(l \in B\). We have \(f_i(x^k) > f_j(x^k)\). Thus,
\[
\alpha_{ik} = \sum_{l \in A} \alpha_{jl} = 1 \cdot b = b.
\]

On the other hand, let
\[
M_1 = \{ l \in B \mid f_i(x^k) < f_j(x^k) \}, \quad m_1 = |M_1|;
\]
\[
M_2 = \{ l \in B \mid f_i(x^k) = f_j(x^k) \}, \quad m_2 = |M_2|;
\]
\[
M_3 = \{ l \in B \mid f_i(x^k) > f_j(x^k) \}, \quad m_3 = |M_3|.
\]

Obviously, \(m_1 + m_2, m_3 \geq 0\) and \(m_1 + m_2 + m_3 = b\). Therefore,
\[
\alpha_{ik} = m_1 \cdot 0 + m_2 \cdot 0.5 + m_3 \cdot 1 \leq b.
\]

- When \(l \in C\). We have
\[
\alpha_{ik} = \sum_{l \in C} \alpha_{jl} = 0.5 \cdot c.
\]

On the other hand, let
\[
M_1 = \{ l \in C \mid f_i(x^k) < f_j(x^k) \}, \quad m_1 = |M_1|;
\]
\[
M_2 = \{ l \in C \mid f_i(x^k) = f_j(x^k) \}, \quad m_2 = |M_2|.
\]

Note that it is not possible to have \(f_i(x^k) > f_j(x^k)\). Otherwise, we will have \(f_j(x^k) > f_i(x^k)\), which is contrary to \(F(x^i) \leq F(x^j)\). Again, we have \(m_1, m_2 \geq 0\) and \(m_1 + m_2 = c\). Thus,
\[
\alpha_{ik} = \sum_{l \in C} \alpha_{jl} = m_1 \cdot 0 + m_2 \cdot 0.5 \leq 0.5c.
\]

In light of Eqs. (3)–(8), we have
\[
\alpha_{ik} = \sum_{a = 1}^{3} \alpha_{ik}^a \geq \alpha_{ik} = \sum_{a = 1}^{3} \alpha_{ik}^a.
\]

**Theorem 3.1.** If \(K_e = \max_{i < N} \{K_i\}\), then the solution \(x^e\) corresponding to \(K_e\) must be an efficient solution (Pareto solution).

**Proof.** If \(x^e\) is not an efficient solution, then there exists an \(x^* \in X\), such that
\[
F(x^e) \leq F(x^*).
\]

Therefore, according to Lemma 3.1,
\[
a_{es} < a_{er}.
\]

Due to Lemma 3.2, we have, for any \(x^i (j \in N, j \neq e, s)\),
\[
a_{ij} \leq a_{ij}.
\]

Based on (9), (10) and \(a_{es} = a_{es} = 0\), we have
\[
K_e = \sum_{j \in N} a_{ej} < \sum_{j \in N} a_{ej} = K_s,
\]

which is a contradiction to the assumption \(K_e = \max_{i < N} \{K_i\}\). The proof is completed. □

Based on Theorem 3.1, the following results are obvious.

**Corollary 3.1.** If \(x^e\) is an absolute optimal solution, then
\[
K_e = \max_{j < N} \{K_j\}.
\]

**Corollary 3.2.** If \(x^e\) is corresponding to \(K_e = \max_{j < N} \{K_j\}\) and \(x^e\) is corresponding to \(K_s = \max_{j < N} \{K_j\}\), then \(x^e\) is an efficient solution of the population without \(x^e\).

Theorem 3.1 and Corollary 3.2 reveal the rationality of optimal sequence method, because the best solution must be an efficient solution, and the second best solution, although may not be an efficient solution, is an efficient solution without considering the best one. This procedure can be continued until the last solution is obtained. It is reasonable for us to rank the population based on their total optimal number and assign fitness according to the ranking index.

### 3.2. Diversity

In order to obtain uniformly distributed solutions, it is important to maintain the diversity of populations. Without diversity-maintaining measures, the population tends to form relatively few clusters in the multi-objective genetic algorithm. This phenomenon is known as genetic drift. Several approaches have been devised to prevent it. Fitness sharing [6,11] encourage the search in unexplored sections of a Pareto frontier by artificially reducing fitness of solutions in densely populated areas. To achieve this goal, densely populated area is identified and a penalty method is incorporated so as to distribute the solutions located in these areas. The difficulty of this method is how to identify the dense areas and measure their density. Crowding distance approach [5] aims to obtain a uniform spread of solutions along the best-known Pareto frontier. The advantage of this approach is that the measurement of population density around a solution can be computed without requiring a user-defined parameter. Cell-based density approach [20,21,30,44] divides the objective space into p-dimensional cells. The number of solutions in each cell is defined as the density of the cell, and the density of a solution is equal to the density of the cell in which the solution locates. An efficient approach to dynamically divide the objective function space into cells is proposed by Lu and Yen [30,44]. The main advantage of this approach is that a global density map of the objective function space is obtained as a result of the density calculation while keeping similar computational efficiency as other approaches.

In this paper, we design a new selection operator based on the combination of the cell-based density approach and the Pareto-ranking approach. This selection operator not only maintains the
density of solutions but also guarantees that all the Pareto solutions are reserved to the next generation.

**Algorithm 3.2.** Cell-based Pareto selection operator.

**Step 1:** Input the current ranked (solutions are firstly sorted by Algorithm 3.1) population $P$ and their corresponding objective function values $F(P)$. Set $\alpha$ as the number of sections for each dimension.

**Step 2:** Separate each dimension of the objective function space into $\alpha$ sections using width

$$w = \frac{z_{\max} - z_{\min}}{\alpha}, \quad k = 1, 2, \ldots, \alpha,$$

where $z_{\max}$ and $z_{\min}$ are the maximum and minimum values of the objective function $k$ in the current population, respectively.

**Step 3:** Classify objective function values $F(P)$ into cells, and measure the density of each cell and the density of each point in $F(P)$.

**Step 4:** Select solutions one by one according to the following substeps until population is fully filled.

**Step 4.1:** Successively check each cell, if the density of a cell is not zero, then choose the first best solution (the order is already sorted by optimal sequence method) in the cell. Let the set $E$ be the collection of all the solutions selected.

**Step 4.2:** Identify Pareto frontier of the set $E$ and collect them in a set denoted by PE.

**Step 4.3:** Maintain the solutions corresponding to points in PE (i.e., $S = \{x \in E|F(x) = y, y \in PE\}$) to the next generation.

**Step 4.4:** Meanwhile, for each point in PE, reduce the density of the cell in which it locates by 1, marking this point as having been selected (which means that it cannot be considered as a candidature again when the loop goes back to Step 4.1).

**Step 4.5:** Check the number of solutions that is kept in the next generation. If the number reaches the population size, then stop and return; otherwise, go to Step 4.1.

**Remark 3.1.** There is only one user-determined parameter $\alpha$ which determines the size of cells in this selection operator. Evidently, a large $\alpha$ leads to a small size of cells. Thus, the enlargement of $\alpha$ will increase the diversity of population, but will lead to a heavy computational burden. For simplicity, we fix the number of sections for all dimensions in the objective function value space, instead of applying different numbers of sections for each dimension [30,44].

**Remark 3.2.** In Step 4.1, the solutions are not chosen strictly according to their optimal number, since those solutions who have high optimal number may locate in a same cell. Instead, only the best solution in that cell can be selected. However, we can guarantee that each solution which is chosen in the corresponding cell is the most reasonable one.

**Remark 3.3.** In Step 4.2, only the efficient points (Pareto points) in $E$ are maintained to the next generation. During the selection process, we not only guarantee diversity of the next generation but also ensure that the solutions selected to be parents of the next generation are better ones.

To illustrate the procedure of Algorithm 3.2, let us investigate a simple example. Fig. 1 shows a population of solutions and their optimal sequence orders assigned by using Algorithm 3.1. The solutions are classified into two dimensional cells. Clearly, some cells include many solutions, such as cells A, B and C, while others include few solutions, such as E, D and F. Then, we select the best point from each cell and collect them constructing a candidature set $E$ which is depicted in Fig. 2. It is easy to note that for those cells who have more than one solutions, only the one has the best optimal number can be selected. For instance, solutions 1, 3 and 6 are selected from cells A, B and C, respectively. Whereas, for those cells who have only one solution, the solution is selected directly, such as solutions 22, 29 and 30 from cells D, E and F, respectively. Finally, Pareto frontier points of set $E$ are identified and maintained to the next generation (see Fig. 3). We can see from the figure that the obtained solutions are not only better solutions but also well distributed.

Suppose that the dimension of the vector function $F$ is $p$, the population size is $N$, then for each component function of $F$, one needs $2N^2$ times of compare, so the computation complexity of Algorithm 3.1 is $O(pN^2)$. In Algorithm 3.2, in order to allocate a cell for one solution, one needs $ap$ times of compare. This is because for each dimension of $F$, we need $\alpha$ times of comparison in the worst case, while the total dimensions of $F$ is $p$. Note that there are $N$ individuals in a population, so the computational complexity of Algorithm 3.2 is $O(\alpha pN)$.

### 3.3. Elitism

Elitism in the single-objective genetic algorithm means that the best individual in a population always survives to the next
generation. In the process of solving multi-objective optimization problems using the genetic algorithm, all the nondominated solutions should be considered as elite solutions. Thus, the number of possible elite solutions becomes very large. The early multi-objective genetic algorithm did not use elitism at all. Since the genetic algorithm with elitist strategies outperforms those without using it, the elitism is incorporated in the recent multi-objective genetic algorithms and their variations [4,42,52]. For these methods, two strategies are introduced to implement elitism [17]: (i) maintaining elitist solutions in the population, and (ii) storing elitist solutions in an external secondary list and reintroducing them to the population by a certain strategy.

Our strategy to implement elitism is simple. We use an extra set, say $B$, to store the best populations we have achieved so far. At each iteration, we put the population obtained from last generation together with $B$ and construct a new set, say $C$. Then, optimal sequence method (Algorithm 3.1) is applied to rank all the solutions in $C$. Finally, we update the best population $B$ by choosing the first population size of best solutions in ranked $C$. This elitism strategy is summarized in the following algorithm.

**Algorithm 3.3.** Optimal sequence elitism.

1. **Step 1:** Input the current population $P$ obtained by the selection operator (Algorithm 3.2); the current best population $B$; the population size.
2. **Step 2:** Let $C = B \cup P$. Apply the optimal sequence method (Algorithm 3.1) on set $C$. Denote the ranked set as $C'$.
3. **Step 3:** Update $B$ by assigning it with the first population size of best solution in $C'$.

**Remark 3.4.** The initial best population $B$ is equal to the initial population.

3.4. **Optimal sequence multi-objective GA (OSMGA)**

Based on the algorithms provided above, we are ready to present the improved multi-objective GA:

**Algorithm 3.4.** Optimal sequence multi-objective GA (OSMGA),

1. **Initialization**

   1.1 Generate the initial population $P(0)$,
   1.2 Initialize the best population $B = P(0)$,

3. **Step 2:** Let $t \rightarrow 0$.
4. **Step 3:** While the maximal generation number is not reached, do

   2.1 **Crossover operator:** generate $O_1(t)$,
   2.2 **Mutation operator:** generate $O_2(t)$,
   2.3 **Evaluate objective function values:** construct selection pool:
   
   $S(t) = O_1(t) \cup O_2(t)$
   
   and compute their multi-objective function values:
   
   $FS(t) = \{f(x) | x \in S(t)\}$
   
   2.4 **Rank solutions in selection pool:** apply optimal sequence method (Algorithm 3.1) to rank the solution in selection pool $S(t)$.
   2.5 **Selection operator**: apply the cell-based Pareto selection operator (Algorithm 3.2) to construct the parents population $P(t \rightarrow t + 1)$.
   2.6 **Elitism:** update the best population $B$ by applying optimal sequence elitism (Algorithm 3.3),

4. **End**

4. **An evaluation system for MOGAs**

In order to show the improvement of a new method, it is necessary to compare the numerical performance of this method with others. However, for the multi-objective genetic algorithm, the numerical performance evaluation is not an evident task, because the multi-objective genetic algorithm gives a population of approximate Pareto solutions, so the comment has to be analyzed from the performances of all the solutions but not any one of them. Therefore, it is needed to develop a reasonable evaluation system for multi-objective genetic algorithms. In this section, we develop an evaluation system to evaluate the numerical performance of different MOGAs.

As it is pointed out by Deb [4], the multi-objective optimization is a mixture of two optimization strategies:

1. minimizing the distance of the obtained solutions from the true optimal solutions.
2. maximizing the spread of solutions in the obtained set of solutions.

The former task is similar to the usual task in a single-objective optimization, while the latter one is similar to finding multiple optimal solutions in a multi-modal optimization. Based on this understanding, we will develop a system to evaluate the distance between the image set of obtained solutions and the real Pareto frontier in the objective function value space; and furthermore, the distributional degree of the obtained solutions.

In the following, we provide two performance metrics which are correspondence to the closeness and diversity of the image set of the obtained solutions. As a matter of convenience, we use $Q$ to represent the solution set obtained from a multi-objective genetic algorithm.

4.1. **Closeness metric**

Veldhuizen [41] introduced a metric called *Error Ratio* which simply counts the number of solutions in $Q$ which does not belong to the Pareto frontier $P, F^*$ and then calculate the ratio of it respect...
to the total number of solutions. Mathematically,

$$\text{ER} = \frac{\sum_{i=1}^{\left|Q\right|} e_i}{\left|Q\right|},$$

(11)

where

$$e_i = \begin{cases} 1 & \text{if } x_i \in Q \text{ and } x_i \in PF^* \\ 0 & \text{if } x_i \in Q \text{ and } x_i \notin PF^* \end{cases}$$

and $\left|Q\right|$ represents the cardinal number of set $Q$. This error ratio metric is simple and evident. However, the requirement of $x_i$ that exactly belongs to $PF^*$ is too strict. From the formula (11), $e_i=0$ even $x_i$ is very close to but not belonging to $PF^*$, which cannot represent the suitable closeness between real Pareto frontier and the image set of the obtained solutions. Although a threshold $\delta$ was introduced in some later literatures, the determination of $\delta$ is still a problem.

In our modified error ratio, we introduce the following function:

$$e_i = e^{-ad_i}, i = 1, 2, \ldots, \left|Q\right|,$$

where $\alpha \in [1, 4]$ is a parameter and $d_i$ is the closest distance between the point and the real Pareto frontier. Mathematically,

$$d_i = \min_{P \in PF^*} d(x_i, P).$$

The distance function $d(\cdot, \cdot)$ could be the Euclidian distance. And eventually, the modified error ratio is still expressed as

$$\text{MER} = \frac{\sum_{i=1}^{\left|Q\right|} e_i}{\left|Q\right|}.$$

Since $d_i \in [0, +\infty)$, $e_i \in [0, 1]$, which makes $\text{MER} \in (0, 1)$. A bigger $\text{MER}$ means the whole solution set is closer to the real Pareto frontier. Note that for some problems, especially those whose optimal Pareto frontier is unknown, the calculation of distance $d_i$ could be difficult, even impossible. However, for the test problems in this paper, their Pareto frontiers are clear.

4.2. Diversity metric

For the diversity of solutions, we expect the solution spread as uniform and extensive as possible. Suppose that we uniformly grid the objective function space into many cubes whose volume is the same. Then it is evident that all the solutions distribute in different cubes. But based on the diversity of population, some cubes may have no solution, some cubes may have just one solution and some cube may have two or more than two solutions. For the cubes having only one solution, we count this solution as an individual one. However, for those cubes having two or more than two solutions we can say that those solutions are very close to each other which is not a good distribution. Thus, we count all of those solutions just as one. Then we calculate the ratio between the number of cubes having solutions and the total number of solutions; and this ratio is used as an evaluation index of the diversity of solutions. The process of this cell-based diversity metric is introduced in detail as follows:

Cell-based diversity metric:

Step 1: Data: input the upper bound ($ub$) and the lower bound ($lb$) of the Pareto frontier; set a parameter $\beta$ as the number of section for each dimension.

Step 2: Grid the objective function space: divide each dimension of objective function space into $\beta$ sections from $lb$ to $ub$. This grids the objective function space into a cell space and then assign each cube an indicator which originally set to be 0.

Step 3: Consecutively check each solution from the population, set the indicator of a cell to be 1 if there is one or more solutions located in the cell.

Step 4: Count the number of cells whose indicator is 1 and assign the number to $M$, then the diversity metric is evaluated as

$$R = \frac{M}{\left|Q\right|}.$$

Remark 4.1. It is evident to note that this diversity metric is sensitively influenced by the number of sections $\beta$. First of all, the grid of objective function value space should not be too fine ($\beta$ is chosen as a large number), otherwise, almost every solution can locate in a different cell which is not good for the evaluation of diversity. On the other hand, the grid should not be too rough ($\beta$ is chosen as a small number) as well, because in this way, many solutions, even they are uniformly distributed may be put into a same cell, consequently, counted as just one solution. It is reasonable to set $\beta$ equal or a little bit larger than the population size. In the way, the ratio $M$ can be guaranteed between 0 and 1.

For a certain number of solutions, it is obvious that a larger $R$ implies that these solutions distribute in more different cells, which means a better diversity is obtained. Fig. 4 illustrates an example of cell-based diversity metric for a hypothetical problem with 20 solutions in total. From the picture, two or more than two solutions distribute in cells A, B, C, D and E, so they are just counted as 5 individuals; plus the other 8 cells who has just 1 solution each. Then, the diversity metric for this set of solutions is $M = 13/20 = 0.65$.

5. Numerical experiments

In this section, we investigate the numerical performance of OSMGA. First, we apply OSMGA on two multi-objective benchmarks quoted from [5]. Then, we compare OSMGA with the methods proposed in CEC’09 [48] using the test problems proposed for CEC’09 [49]. All the numerical experiments are implemented in an environment of MATLAB(2010a) installed on an ACER ASPIRE 4730Z laptop with a 2 G RAM and a 2.16 GB CPU.
5.1. Numerical performance of OSMGA

The test problems (see Table 2) applied in this subsection are quoted from [5]. Problem SCH is a one-dimensional convex problem, its Pareto solutions are \( x \in [0, 2] \). Problem FON is a three-dimensional nonconvex problem, its Pareto solutions satisfy \( x_1 = x_2 = x_3 \), where \( x_i \in [-1/\sqrt{3}, 1/\sqrt{3}], i = 1, 2, 3 \). Fig. 5 depicts the objective function value space of Problems SCH and FON (the red curves represent Pareto frontiers). From the figure, one can observe that their Pareto frontiers are connected.

We solve the proposed benchmarks by OSMGA and other MOGAs, such as VEGA [35], NSGA [38], NSGAII [5], and NPGA [15]. The results are compared using the evaluation system developed in the previous section. In the implementation of OSMGA, parameters are adjusted according to the scale of problem. Empirically, if the dimension of the problem is \( n \), then the population size is \( 2n–5n \), the number of maximal generation is \( 20n–50n \), the crossover rate and the mutation rate are \( 0.4–0.5 \) and \( 0.2–0.3 \), respectively. In Algorithm 3.1, we use \( 10–15 \) for the number of sections in each dimension. In the following tables, the following notations are used:

- MER—the modified error ratio of Pareto solutions;
- ave. MER—the average of modified error ratio;
- std. MER—the standard deviation of modified error ratio;
- \( R \)—the diversity metric of Pareto solutions;
- ave. \( R \)—the average of diversity metric;
- std. \( R \)—the standard deviation of diversity metric.

For each problem, we run OSMGA independently for 100 times. Table 3 demonstrates the statistic results of the experiments. From the table, the average of modified error ratio is very close to 1 and its standard deviation is almost equal to 0, which reveals that for the considered benchmarks, the approximate Pareto frontier obtained by OSMGA is steadily located on the real Pareto frontier. The average and the standard deviation of the diversity metric of Problem SCH’s solutions are 0.50 and 0.11, respectively, which are not very good; however, for Problem FON, they are 0.83 and 0.08, respectively, which are very promising.

Fig. 6 depicts some intermediate results of Problems SCH and FON solved by OSMGA. From the figures, we can observe that OSMGA converges very fast in solving these two problems. For Problem SCH, the range of Pareto frontier is located at the third generation and the implementation terminates at the tenth generation. For Problem FON, the range of Pareto frontier is located at the 19th generation and the simulation terminates at the 30th generation. The obtained approximate Pareto frontier for both problems is accurate and uniformly distributed.

Table 4 illustrates numerical results of Problem SCH solved by OSMGA, VEGA, NSGA, NSGAII and NPGA for 10 independence implementations. From the table, one can observe that the approximate Pareto frontiers obtained by different methods are all very close to the real Pareto frontier. However, the diversity
metric of OSMGA is evidently larger than the other methods. It is not uncommon to see that diversity metric of VEGA is lower than the other four methods. Actually, VEGA, as the first multi-objective genetic algorithm, did not take account of diversity. The method of NSGAII, although not very stable, enjoys a modest diversity metric. Fig. 7(a) depicts the best implementation for each methods. We can observe that the approximate Pareto solutions obtained by these methods are close to the real Pareto frontier. This fits with the data in Table 4. However, solutions obtained by VEGA, NSGA and NPGA are very concentrative, while solutions obtained by OSMGA and NSGAII are better distributed.

Problem FON is a nonconvex multi-objective optimization problem whose Pareto solutions satisfy $x_1 = x_2 = x_3 \in [-1/\sqrt{3}, 1/\sqrt{3}]$. Table 5 illustrates 10 independent implementations of OSMGA, VEGA, NSGA, NSGAII and NPGA on Problem FON. From the table, we can observe that solutions obtained by OSMGA,
NSGAII and NPGA are all very close to real Pareto frontier, some of the solutions obtained by VEGA are not very good, the solutions obtained by NSGA are failed. For the diversity metric, comparison among OSMGA, NSGAII and NPGA, OSMGA enjoys the best performance, the solutions obtained by OSMGA are well distributed along the Pareto frontier. Whereas, most of the solutions obtained by NSGAII distribute on the boundary and solutions obtained by NPGA distribute in the middle of the real Pareto frontier but very close to each other. Fig. 7(b) shows the best performance for each method among the 10 implementations. We can observe that the figure is identical with the table data.

### 5.2. Numerical comparison

In this subsection, we compare the numerical performance of OSMGA with the methods proposed in the special session on performance assessment of unconstrained/bound constrained multi-objective optimization algorithms at CEC’09. There are 13 algorithms submitted to the special session:

- MOEAD [47];
- GDE3 [23];
- MOEADGM [1];
- MTS [40];
- LiuLiAlgorithm [26];
- DMOEADD [27];
- NSGAIILS [37];
- OWMOSaDE [16];
- ClusteringMOEA [43];
- AMGA [39];
- MOEP [33];
- DECMOSA-SQP [45];
- OMOEAII [7].

The test problems applied in this subsection are quoted from [49] and these are used as benchmarks in CEC’09. Fig. 8 illustrates the objective function value space of these test problems (the figure of test problem 2 is ignored here since it is similar to that of the test problem 1), the red curve/surface represents their Pareto frontiers. Among these test problems, Problems 1–7 have two objective functions, whereas Problems 8–10 have three objective functions. The Pareto solutions of Problems 5, 6 and 9 are disconnected, while the others are connected.

In order to evaluate the numerical performance, we use the performance metric IGD proposed in [49]. Suppose that \( P^* \) is a set

**Table 5**

Comparison among VEGA, NSGA, NSGAII and NPGA on problem FON.

<table>
<thead>
<tr>
<th>Implementation index</th>
<th>OSMGA</th>
<th>VEGA</th>
<th>NSGA</th>
<th>NSGAII</th>
<th>NPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.37</td>
<td>0.79</td>
<td>0.04</td>
<td>0.84</td>
</tr>
<tr>
<td>2</td>
<td>0.99</td>
<td>0.35</td>
<td>0.76</td>
<td>0.02</td>
<td>0.81</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>0.30</td>
<td>0.69</td>
<td>0.02</td>
<td>0.74</td>
</tr>
<tr>
<td>4</td>
<td>0.99</td>
<td>0.25</td>
<td>0.70</td>
<td>0.04</td>
<td>0.85</td>
</tr>
<tr>
<td>5</td>
<td>0.99</td>
<td>0.30</td>
<td>0.90</td>
<td>0.02</td>
<td>0.69</td>
</tr>
<tr>
<td>6</td>
<td>0.99</td>
<td>0.53</td>
<td>0.89</td>
<td>0.04</td>
<td>0.88</td>
</tr>
<tr>
<td>7</td>
<td>0.99</td>
<td>0.30</td>
<td>0.88</td>
<td>0.04</td>
<td>0.75</td>
</tr>
<tr>
<td>8</td>
<td>0.99</td>
<td>0.50</td>
<td>0.89</td>
<td>0.04</td>
<td>0.79</td>
</tr>
<tr>
<td>9</td>
<td>0.99</td>
<td>0.28</td>
<td>0.89</td>
<td>0.02</td>
<td>0.89</td>
</tr>
<tr>
<td>10</td>
<td>0.99</td>
<td>0.34</td>
<td>0.89</td>
<td>0.04</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Fig. 8. The objective function value space for test problems.
of uniformly distributed points along the Pareto frontier (in the objective function value space). Let \( A \) be a set of solutions obtained by a certain solver. Then, the average distance from \( P_n \) to \( A \) is defined as

\[
IGD(A, P_n) = \frac{\sum_{v \in P_n} d(v, A)}{|P_n|}
\]

where \( d(v, A) \) is the minimum Euclidean distance between \( v \) and the points in \( A \), i.e.,

\[
d(v, A) = \min_{y \in A} ||v - y||
\]

In fact, \( P_n \) represents a sample set of the real Pareto frontier, if \( |P_n| \) is large enough to approximate the Pareto frontier very well.

Table 6
The numerical performance evaluated by IGD.

<table>
<thead>
<tr>
<th>Rank</th>
<th>UF1</th>
<th>UF2</th>
<th>UF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOEAD</td>
<td>0.00435</td>
<td>MTS</td>
</tr>
<tr>
<td>2</td>
<td>GDE3</td>
<td>0.00534</td>
<td>MOEADGM</td>
</tr>
<tr>
<td>3</td>
<td>MOEADGM</td>
<td>0.0062</td>
<td>DMOEADD</td>
</tr>
<tr>
<td>4</td>
<td>MTS</td>
<td>0.00646</td>
<td>MTS</td>
</tr>
<tr>
<td>5</td>
<td>LiuLiAlgorithm</td>
<td>0.00785</td>
<td>OWSMOSaDE</td>
</tr>
<tr>
<td>6</td>
<td>DMOEADD</td>
<td>0.01038</td>
<td>GDE3</td>
</tr>
<tr>
<td>7</td>
<td>NSGAII</td>
<td>0.01153</td>
<td>LiuLiAlgorithm</td>
</tr>
<tr>
<td>8</td>
<td>OWSMOSaDE</td>
<td>0.0122</td>
<td>NSGAII</td>
</tr>
<tr>
<td>9</td>
<td>OWSMOSaDE</td>
<td>0.0122</td>
<td>NSGAII</td>
</tr>
<tr>
<td>10</td>
<td>ClusteringMOEA</td>
<td>0.0299</td>
<td>OSMGA</td>
</tr>
<tr>
<td>11</td>
<td>AMGA</td>
<td>0.03588</td>
<td>MOEP</td>
</tr>
<tr>
<td>12</td>
<td>MOEP</td>
<td>0.0596</td>
<td>ClusteringMOEA</td>
</tr>
<tr>
<td>13</td>
<td>DECMSOSA-SQP</td>
<td>0.07702</td>
<td>OSMGA</td>
</tr>
<tr>
<td>14</td>
<td>OMOEAI</td>
<td>0.08564</td>
<td>OMOEAI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>UF4</th>
<th>UF5</th>
<th>UF6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MTS</td>
<td>0.02356</td>
<td>MTS</td>
</tr>
<tr>
<td>2</td>
<td>GDE3</td>
<td>0.0265</td>
<td>AMGA</td>
</tr>
<tr>
<td>3</td>
<td>DECMSOSA-SQP</td>
<td>0.03392</td>
<td>DECMSOSA-SQP</td>
</tr>
<tr>
<td>4</td>
<td>AMGA</td>
<td>0.04062</td>
<td>LiuLiAlgorithm</td>
</tr>
<tr>
<td>5</td>
<td>DMOEADD</td>
<td>0.04268</td>
<td>DECMSOSA-SQP</td>
</tr>
<tr>
<td>6</td>
<td>MOEP</td>
<td>0.0427</td>
<td>OMSGA</td>
</tr>
<tr>
<td>7</td>
<td>LiuLiAlgorithm</td>
<td>0.0435</td>
<td>MOEAD</td>
</tr>
<tr>
<td>8</td>
<td>OMOEAI</td>
<td>0.04624</td>
<td>MOEP</td>
</tr>
<tr>
<td>9</td>
<td>MOEADGM</td>
<td>0.0476</td>
<td>ClusteringMOEA</td>
</tr>
<tr>
<td>10</td>
<td>OWSMOSaDE</td>
<td>0.0513</td>
<td>DMOEADD</td>
</tr>
<tr>
<td>11</td>
<td>OSMGA</td>
<td>0.05392</td>
<td>OWSMOSaDE</td>
</tr>
<tr>
<td>12</td>
<td>NSGAII</td>
<td>0.0584</td>
<td>NSGAII</td>
</tr>
<tr>
<td>13</td>
<td>ClusteringMOEA</td>
<td>0.0585</td>
<td>OSMGA</td>
</tr>
<tr>
<td>14</td>
<td>MOEAD</td>
<td>0.06385</td>
<td>MTS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>UF7</th>
<th>UF8</th>
<th>UF9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOEAD</td>
<td>0.00444</td>
<td>MTS</td>
</tr>
<tr>
<td>2</td>
<td>LiuLiAlgorithm</td>
<td>0.0077</td>
<td>MTS</td>
</tr>
<tr>
<td>3</td>
<td>MOEADGM</td>
<td>0.0076</td>
<td>LiuLiAlgorithm</td>
</tr>
<tr>
<td>4</td>
<td>DMOEADD</td>
<td>0.01032</td>
<td>LiuLiAlgorithm</td>
</tr>
<tr>
<td>5</td>
<td>MOEP</td>
<td>0.0197</td>
<td>OWSMOSaDE</td>
</tr>
<tr>
<td>6</td>
<td>NSGAII</td>
<td>0.02132</td>
<td>MTS</td>
</tr>
<tr>
<td>7</td>
<td>ClusteringMOEA</td>
<td>0.0223</td>
<td>AMGA</td>
</tr>
<tr>
<td>8</td>
<td>DECMSOSA-SQP</td>
<td>0.02416</td>
<td>OMSGA</td>
</tr>
<tr>
<td>9</td>
<td>GDE3</td>
<td>0.02522</td>
<td>OSMGA</td>
</tr>
<tr>
<td>10</td>
<td>OMOEAI</td>
<td>0.03354</td>
<td>OSMGA</td>
</tr>
<tr>
<td>11</td>
<td>MTS</td>
<td>0.04079</td>
<td>DECMSOSA-SQP</td>
</tr>
<tr>
<td>12</td>
<td>AMGA</td>
<td>0.05707</td>
<td>ClusteringMOEA</td>
</tr>
<tr>
<td>13</td>
<td>OWSMOSaDE</td>
<td>0.0585</td>
<td>GDE3</td>
</tr>
<tr>
<td>14</td>
<td>OSMGA</td>
<td>0.13056</td>
<td>MOEP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>UF10</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MTS</td>
<td>0.15306</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>OSMGA</td>
<td>0.24821</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DMOEADD</td>
<td>0.32211</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AMGA</td>
<td>0.32418</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MOEP</td>
<td>0.3621</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DECMSOSA-SQP</td>
<td>0.36985</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ClusteringMOEA</td>
<td>0.4111</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GDE3</td>
<td>0.43326</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LiuLiAlgorithm</td>
<td>0.44691</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>MOEAD</td>
<td>0.47415</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>MTS</td>
<td>0.5646</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>OMOEAI</td>
<td>0.62754</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>OWSMOSaDE</td>
<td>0.743</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>NSGAII</td>
<td>0.84468</td>
<td></td>
</tr>
</tbody>
</table>
IGD\((A, P^*)\) could measure both the diversity and convergence of \(A\) in a sense. A smaller IGD\((A, P^*)\) means the set \(A\) is closer to the real Pareto frontier and has better diversity.

In order to keep consistent with the final report of CEC’09 [48], in the implementation of OOMOGA, we set the population size as 100 for problems with two objectives and 150 for problems with three objectives, the number of function evaluations is less than 300,000. For each test instance, we run OOMOGA independently for 30 times. The numerical performance evaluated by IGD are illustrated in Table 6.

From Table 6, the proposed method OSMGA performs the best at solving Problem 9; its IGD evaluation is 0.04762, better than all the other solvers. In solving Problem 10, OSMGA (whose IGD evaluation is 0.24821) performs only worse than MTS (whose IGD evaluation is 0.15306) but better than all the other solvers. In solving Problem 3, the IGD evaluation of OSMGA (0.02241) is ranked thirdly, worse than MOEAD (0.0072) and LiuLiAlgorithm (0.01497). The numerical performance of OSMGA is moderate in solving Problems 1, 2, 4, 6 and 8, the IGD ranks are 9, 10, 11, 9 and 9, respectively. For Problems 5 and 7, the numerical performance of OSMGA is not so good, ranking in 13 and 14, respectively.

It is not uncommon that the numerical performance of the proposed solver OSMGA is unstable among different test problems. Because the numerical results are not only affected by the performance of solvers, but also impacted by the linearity and structure of objective function themselves. Furthermore, the crossover and mutation operators are also affected by the distribution of points in the objective function value space. Generally speaking, if a new point generated by the crossover or mutation operators has a higher probability of locating around the Pareto frontier, then the Pareto frontier can be well approximated by the solver, for example, Problems 3, 4 and 9. On the contrary, if it is hard for the crossover or mutation operators to generate new point around the Pareto frontier, then the problem is difficult to be solved by MOGAs, for instance, Problems 5, 6 and 7. In fact, this instability still appears in the other solvers, such as MOEAD, which is reported as the best solver in [48]. The IGD evaluation of MOEAD in solving Problems 4, 5 and 10 is not very good, ranking in 14, 7 and 10, respectively.

Fig. 9 demonstrates Pareto frontiers of Problems 1, 2, 4 and 9, respectively. From Fig. 9(a) and (b), we can observe that for Problems 1 and 2, the proposed solver obtained very good representations of their Pareto frontiers. Results for Problem 4 (see Fig. 9(c)) are not very uniformly distributed, which may affect the performance of IGD evaluation. Fig. 9(d) illustrates the approximate Pareto frontier of Problem 9, we can see that the structure is more or less an approximation of the real Pareto frontier.

6. Conclusion

In this paper, we proposed an improved genetic algorithm for solving multi-objective optimization problems. The proposed method is abbreviated as OSMGA. In each iteration, the population is first ranked by the optimal sequence method. The fitness of each solution is assigned using the rank. The diversity of a population is
measured using a method combining Pareto-based ranking and cell-based diversity. Then, parents for the next generation are selected according to their fitness value and diversity measurement. In the proposed method, the best population is updated at each iteration such that the image of solutions is diversely distributed along the real Pareto frontier. In order to compare the numerical performance of different multi-objective genetic algorithms, we developed an evaluation system which takes account of the closeness metric and diversity metric. We tested OSMGA by the two well-known multi-objective benchmarks, results show that OSMGA is promising in solving small scale convex multi-objective optimization problems. We also applied OSMGA on the famous test instances proposed in CEC09 and compared its numerical results with the methods proposed in CEC09. The rank of OSMGA shows that it is efficient and robust in solving unconstrained multi-objective optimization problems.

Acknowledgments

Changzhi Wu was partially supported by Australian Research Council Linkage Program (Grant No. LP140100873), Natural Science Foundation of China (61473326), Natural Science Foundation of Chongqing (cstc2013jcyjA00029 and cstc2013jB0149).

This publication was made possible by NPRP Grant # NPRP 4—1162—1—181 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

References


