Abstract—Many real world applications require optimizing multiple objectives simultaneously. Multiobjective evolutionary algorithm based on decomposition (MOEA/D) is a new framework for dealing with such kind of multiobjective optimization problems (MOPs). MOEA/D focuses on how to maintain a set of scalarized sub-problems to approach the optimum of a MOP. This paper addresses the offspring reproduction operator in MOEA/D. It is arguable that, to design efficient offspring generators, the properties of both the algorithm to use and the problem to tackle should be considered. To illustrate this idea, a generator based on multivariate Gaussian models is proposed under the MOEA/D framework in this paper. In the new generator, both the local and global population distribution information is extracted by a set of Gaussian distribution models; new trial solutions are sampled from the probability models. The proposed approach is applied to a set of benchmark problems with complicated Pareto sets. The comparison study shows that the offspring generator is promising for dealing with continuous MOPs.

Index Terms—Multiobjective evolutionary algorithm, decomposition, probabilistic model.

I. INTRODUCTION

This paper considers the following continuous multiobjective optimization problem (MOP):

$$\begin{align*}
\text{min} & \quad F(x) = (f_1(x), \ldots, f_m(x))^T \\
\text{s.t} & \quad x \in \Pi_{i=1}^n [a_i, b_i] \\
\end{align*}$$

(1)

where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is a decision variable vector, $\Pi_{i=1}^n [a_i, b_i] \subset \mathbb{R}^n$ is the feasible search space, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \ldots, m$, is a continuous mapping, and $F(x)$ is an objective vector.

Since the multiple objectives usually contradict each other, there does not exist an optimal solution that can minimize all the objectives in (1) at the same time. Therefore, one needs to balance these objectives. Pareto optimal solutions are candidates for the best tradeoffs among the objectives. The Pareto set (PS) is the set of all the Pareto optimal solutions and the Pareto front (PF) is the image of the PS in the objective space.

Among different methods, evolutionary algorithms (EAs), which aim to find a good approximation to the PS (PF), are promising for dealing with MOPs [1]. A number of multiobjective evolutionary algorithms (MOEAs) have been proposed [2]. Most current MOEAs, such as PAES [3], NSGA-II [4], SPEA2 [5], push the population as a whole towards the PF, guided mainly by dominance-based fitness measures that rely on long chains of pair-wise comparisons. Multiobjective evolutionary algorithm based on Decomposition (MOEA/D) [6], [7] works in a different way. MOEA/D decomposes a MOP into a set of scalar objective optimization problems (sub-problems) and the optimal solution of each subproblem will hopefully be a Pareto optimal solution point. MOEA/D solves these sub-problems simultaneously. Quite a few variants of MOEA/D algorithms have been reported for both continuous and discrete problems [8]–[11].

Estimation of distribution algorithms (EDAs) [12] are a special type of EAs. Unlike traditional EAs, which use crossover and/or mutation operators to generate new trial solutions, EDAs explicitly extract the population distribution information by probabilistic models and sample new trial solutions from the models thus built. In offspring reproduction, EAs use individual information, while EDAs use global distribution information. In practice, it is hard to say which strategy is better. Therefore, it might be better to combine the two information [13]. How to select a proper probability distribution to model the population information is one of the key issues in EDAs. Usually, the univariate models, bivariate models, multivariate models, and mixture models are used in EDAs [14]. Since complicated models may represent the population more reasonably and accurately than simple models, they are more likely to be used to design EDAs. However, the target of EDAs is doing optimization instead of building good models, although the optimization results highly depend on the model quality. A proper model should consider the properties of both the algorithm to use and the problems to tackle. To achieve this goal, the statistical and machine learning techniques, such as dimension reduction, regression, density estimation, etc., could help to build such a proper model in EDAs.

The multivariate Gaussian distribution model might be the most popular probability models used in EDAs. It has been widely studied for both scalar objective and multiobjective optimization [14]–[20]. In this paper, we propose to combine multivariate Gaussian models and MOEA/D for continuous multiobjective optimization. By considering the property of the MOEA/D framework, a Gaussian distribution model is built either around a sub-problem or based on the whole population.
in MOEA/D. All the probability models could thus capture the population distribution from a global view.

The rest of the paper is organized as follows. In Section II, we present the proposed algorithm with details. The algorithm framework and reproduction operator are addressed. Section III provides the experimental results as well as the analysis. Finally, the paper is concluded in Section IV.

II. MOEA/D WITH MULTIVARIATE GAUSSIAN MODELS

MOEA/D decomposes a MOP into a set of sub-problems \( \{g^1, g^2, \cdots, g^N\} \) and solves them simultaneously. The optimal solution of each sub-problem will hopefully be a Pareto optimal solution to the MOP under consideration. Solving a set of well selected sub-problems may produce a good approximation to the PS (PF). To solve the sub-problems in a collaborative way, MOEA/D defines neighborhood relationship among all the sub-problems based on their similarity. A sub-problem is optimized mainly using information from its neighboring sub-problems. Fig. 1 illustrates the basic idea of MOEA/D for bi-objective optimization.

\[
\min g^i(x) = g(x|\lambda^i, z^* \cap \pi) = \max_{1 \leq j \leq m} \lambda^i_j |f_j(x) - z^*_j| \quad (2)
\]

where \( \lambda^i = (\lambda^i_1, \cdots, \lambda^i_m)^T \) is a weight vector, \( z^* = (z^*_1, \cdots, z^*_m)^T \) is a reference point, i.e., \( z^*_j \) is the minimal value of \( f_j \) in the search space. In most cases, two sub-problems with close weight vectors will have similar optimal solutions. Based on the distances among the weight vectors, MOEA/D defines neighborhood relations among the sub-problems.

In MOEA/D, the following information of the \( i \)th (\( i = 1, \cdots, N \)) sub-problem should be maintained:

- its objective function \( g^i(x) \) in (2),
- its current solution \( x^i \) and the objective vector of \( x^i \), i.e. \( F^i = F(x^i) \), and
- the index set of its neighboring sub-problems, \( B^i \).

Thus, we could use a tuple of \( (x^i, F^i, B^i, g^i) \), to denote the \( i \)th sub-problem. MOEA/D also need to maintain:

- a reference point \( z^* = (z^*_1, \cdots, z^*_m)^T \).

The main framework of MOEA/D is as follows.

**Algorithm 1: Main Framework of MOEA/D**

1. Initialize a set of sub-problems \( \{x^i, F^i, B^i, g^i\} \), \( i = 1, \cdots, N \);
2. Initialize the reference point \( z^* \) as \( z^*_j = \min_{i=1,\cdots,N} f_j(x^i), j = 1, \cdots, m \);
3. while not terminate do
   4. foreach \( i \in \text{perm}(\{1, \cdots, N\}) \) do
      5. if \( \text{rand()} < p_n \) then
         6. \( \pi = \text{perm}(B^i) \);
      7. else
         8. \( \pi = \text{perm}(\{1, \cdots, N\}) \);
      9. end
   10. Generate a trial point \( y \) by using solutions with indices in \( \pi \);
   11. foreach \( j \in \{1, \cdots, m\} \) do
      12. if \( f_j(y) < z^*_j \) then
         13. Set \( z^*_j = f_j(y) \);
      14. end
   15. end
   16. Set counter \( c = 0 \);
   17. foreach \( j \in \pi \) do
      18. if \( g^j(y) < g^j(x^i) \) and \( c < C \) then
         19. Replace \( x^i \) by \( y \);
         20. Set \( c = c + 1 \);
      21. end
   22. end
23. end
24. end

We would make the following comments on the MOEA/D framework:

- \( N \) is the number of sub-problems.
- \( K \) is the neighborhood size for each sub-problem.
- \( p_n \) is an algorithm parameter which balance the exploitation and exploration of the algorithm.
- \( C \) is the maximal number of old solutions which are allowed to be replaced by a new one.
- \( \text{perm}(\cdot) \) randomly permutes the input values, and \( \text{rand}() \) generates a random real number in \([0, 1]\).
• Line 1: The initial solutions for the sub-problems are uniformly randomly sampled from the feasible space. The weight vectors with the sub-problems are uniformly distributed. The details on generating these weights are referred to [7].
• Line 3: A maximum number of generations is used as the termination condition.
• Line 4: In each iteration, a sub-problem is randomly selected.
• Lines 5-9: The parent set \( \pi \) is either the neighborhood or the whole population which is controlled by \( p_n \). If the neighborhood is used, the algorithm will do exploitation; otherwise, it will do exploration.
• Lines 11-15: The reference point \( z^* \) is updated by the newly generated solution. Since \( z^* \) is not fixed, the sub-problem objectives \( g^i, i = 1, \cdots, N \), are changing during the run.
• Lines 16-23: The parents are updated by the new trial solution. To prevent premature convergence, at most \( C \) parents could be updated. The update is based on the sub-problem objective values.

More details about the MOEA/D framework and its variants could be found in [7]. In the next section, we address the reproduction procedure as listed in Line 10 of Algorithm 1.

B. Multivariate Gaussian Model based Reproduction

Under mild conditions, the PS (PF) of a continuous MOP with \( m \) objective forms an \((m - 1)\) dimensional piecewise continuous manifold in the decision space (objective space). This regularity property of continuous MOPs has been used in designing RM-MEDA recently [21]. An efficient MOEA for continuous MOPs should consider or use this property in an explicit or implicit way. Since a multivariate Gaussian model is centered in a point (not a manifold) in the search space, it is not suitable to using a single Gaussian model to model the population distribution in MOEAs.

In MOEA/D, neighboring solutions are close to each other. Therefore, we can use a single multivariate Gaussian model to describe the distribution of the neighbor points, i.e., we can assume that the neighboring solutions \( \{x^j\}, j \in B^i \), are sampled from

\[
x \sim N(\mu^i, \Sigma^i)
\]

where \( \mu^i \) is an \( n \) dimensional mean vector, \( \Sigma^i = [\text{Cov}(x_s, x_t)] \), \( s, t = 1, \cdots, n \), is an \( n \times n \) covariance matrix, and \( \text{Cov}(x_s, x_t) \) denotes the covariance between the elements \( x_s \) and \( x_t \) of a variable vector.

In our proposed EDA approach in the MOEA/D framework, a set of solutions (parents) from \( \pi \) (Line 10) are used to build a Gaussian model. When \( \pi = B \), all the \( K \) neighbor solutions are used; otherwise when \( \pi = \{1, \cdots, N\} \), \( K \) randomly selected parents are used. A trial solution \( y \) is then sampled from this model. Clearly, when \( \pi = \{1, \cdots, N\} \), such a sampling method can do exploration. Otherwise, it will be mainly for exploitation.

1) Mean vector estimation: When we build a Gaussian model in Line 10 for generating a trial solution. It is natural to assume that the mean of the Gaussian model is the solution of the \( i \)th sub-problem which is under consideration. i.e.,

\[
\mu^i = x^i.
\]

2) Covariance matrix estimation: In this paper, we use the unbiased sample covariance:

\[
\text{Cov}(x_s, x_t) = \frac{1}{K-1} \sum_{j=1}^{K} (x^j_s - x^j_t)^2,
\]

for \( s, t = 1, \cdots, n \).

3) Trial solution sampling and mutating: For the \( i \)th sub-problem, a multivariate Gaussian model is built and a new trial solution is then sampled. To improve the search ability, a small variation is added to the sampled point. The polynomial mutation [1] is used for this purpose. The whole procedure works as follows.

- Decompose the covariance matrix \( \Sigma^i \) into a lower triangular matrix \( A^i \), \( \Sigma^i = A^i A^T \), by using the Cholesky decomposition method.
- Sample a vector \( S = (s_1, \cdots, s_n)^T \), of which the element \( s_j, j = 1, \cdots, n \), is sampled from a standard Gaussian distribution \( N(0, 1) \).
- Generate the new trial solution \( y^i \) as \( y^i = \mu^i + A S \).
- Mutate the new trial solution \( y^i \) by the polynomial mutation operator.

III. EXPERIMENTAL RESULTS

A. Experimental Settings

We call the proposed MOEA/D with multivariate Gaussian models MOEA/D-MG. In this section, we apply MOEA/D-MG to nine test instances with complicates PS shapes which were introduced in [7]. We also compare MOEA/D-MG with MOEA/D-DE [7], in which a differential evolution operator and the polynomial mutation operator were used to genera new trial solutions. The details of the test problems and MOEA/D-DE could be found in [7].

The parameter settings are as follows.

- The number of decision variables are \( n = 30 \) for all the test instances. Both algorithms stop after 500 generations.
- The statistical results are based on 30 independent runs.
- The population size, i.e., the number of sub-problems, is 300 for bi-objective problems and 595 for tri-objective problems.
- Both the two algorithms are implemented in Matlab and run in a same desktop computer.
- In MOEA/D-DE, the other parameters for MOEA/D framework are: \( K = 20 \), \( p_n = 0.9 \), \( C = 2 \), \( CR = 1.0 \), and \( F = 0.5 \) in the DE operator, and \( \eta = 20 \) and \( p_m = \frac{1}{n} \) in the polynomial mutation operator. All these parameters are the same as in [7].
- In MOEA/D-GM, the other parameters for MOEA/D framework are: \( K = 15 \) for bi-objective problems and \( K = 30 \) for tri-objective problems, \( p_n = 0.8 \), \( C = 2 \). The
TABLE I
THE STATISTICAL RESULTS ON THE FINAL PF APPROXIMATIONS OBTAINED BY MOEA/D-DE AND MOEA/D-MG ON THE SIX TEST INSTANCES OVER 30 RUNS.

<table>
<thead>
<tr>
<th>instance</th>
<th>MOEA/D-DE</th>
<th>MOEA/D-MG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std</td>
</tr>
<tr>
<td>$F_1$</td>
<td>0.0008</td>
<td>0.0000</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0.0029</td>
<td>0.0008</td>
</tr>
<tr>
<td>$F_3$</td>
<td>0.0028</td>
<td>0.0016</td>
</tr>
<tr>
<td>$F_4$</td>
<td>0.0070</td>
<td>0.0056</td>
</tr>
<tr>
<td>$F_5$</td>
<td>0.0060</td>
<td>0.0022</td>
</tr>
<tr>
<td>$F_6$</td>
<td>0.3443</td>
<td>0.1761</td>
</tr>
<tr>
<td>$F_7$</td>
<td>0.1885</td>
<td>0.1367</td>
</tr>
<tr>
<td>$F_8$</td>
<td>0.0213</td>
<td>0.0138</td>
</tr>
<tr>
<td>$F_9$</td>
<td>0.0040</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Fig. 2. The mean IGD values versus generations obtained by the two algorithms over 30 runs. The solid lines are with MOEA/D-DE and the dash lines are with MOEA/D-MG.
Fig. 3. The best PF approximations and corresponding PS approximations obtained by MOEA/D-MG according to the IGD values over 30 runs.
Fig. 4. All the final 30 PF and PS approximations obtained by MOEA/D-MG.
control parameters in the polynomial mutation operator are $\eta = 20$ and $p_m = \frac{1}{n}$.

B. Performance Metric

The Inverted Generational Distance (IGD) metric [21], [22] is used to assess the algorithm performances in our experimental studies. Let $P^*$ be a set of uniformly distributed Pareto optimal points in the PF. Let $P$ be an approximation to the PF. The IGD metric is defined as follows,

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|}$$

where $d(v, P)$ is a minimum distance between $v$ and any point in $P$ and $|P^*|$ is the cardinality of $P^*$. The IGD metric can measure both convergence and diversity. To have a low IGD value, $P$ must be close to the PF and cannot miss any part of the whole PF. In our experiments, 500 evenly distributed points in PF are generated as the $P^*$ for bi-objective problems and 990 points for tri-objective problems.

C. Comparison Results and Analysis

Firstly, we compare the performances of MOEA/D-MG and MOEA/D-DE. The statistical results of the IGD metric are shown in Table I. The average IGD values vs. the numbers of generation for these two algorithms are plotted in Fig. 2. Figs. 3 and 4 plot the best and all the approximations obtained by MOEA/D-MG respectively.

The statistical results in Table I are based on the final PF approximations obtained by the algorithms. From this table, it is evident that MOEA/D-MG performs slightly better than MOEA/D-DE on six instances and slightly worse than MOEA/D-DE on the three other instances.

Fig. 2 shows the run time performance. It is clear that convergence speeds of the two algorithms are more or less similar on F1-F5, F8 and F9. MOEA/D-MG beats MOEA/D-DE on F6 and F7.

The test instances are with complicated PS shapes as shown in Figs. 3 and 4. Fig 3 plots and the best approximations obtained by MOEA/D-MG. Clearly, they can approximate both the PFs and PSs well on all the instances except F5. On F5, a small part of PS corresponds to a large part of PF. MOEA/D-MG fails to approximate the whole PS. The reason might be that MOEA/D-MG could not build high quality models. All the 30 final approximations in Fig. 4 indicate that MOEA/D-MG can cover the whole PFs and PSs on F1-F6 but failed on F7-F9. F7 and F8 have many local PFs, MOEA/D-MG may be trapped in some locally optimal PSs in some runs. F9 has a concave PF, MOEA/D-MG can not approximate the left part of the PF. The reason might be that the sub-problems are dense in this area although the weight vectors are well distributed.

D. Sensitivity of Algorithm Parameters

In this section, we study the sensitivity of the neighborhood size $K$ and neighborhood search probability $p_n$, which are the major control parameters in MOEA/D-MG. We test $K = 10, 15, 20, 25, 30, 35, 40$ and $p_n = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$ on F1-F5. The final mean IGD values versus the control parameters are plotted in Fig. 5.

Fig. 5 shows that a small neighborhood size is preferable for MOEA/D-MG. Although as $K$ increases, the mean IGD values also increase but they do not change much. When $p_n = 1.0$, no exploration is applied and the performances deteriorate on F1, F2 and F3 as illustrated in Fig. 5. However, when $p_n$ changes from 0.5 to 0.9, the performances do not change much. The results show that exploration is necessary for MOEA/D-MG.

IV. CONCLUSIONS

In this paper, we proposed an alternative offspring reproduction operator for MOEA/D based on multivariate Gaussian distributions. The basic idea is to build a multivariate Gaussian model around a solution in MOEA/D. A new trial solution is sampled from the Gaussian model thus built and mutated by the polynomial mutation operator. It then updates the neighbor solutions. The new algorithm, MOEA/D-MG, was tested on a set of test instances with complicated PS shapes. The statistical results show that MOEA/D-MG is comparable with MOEA/D-DE. Both the two MOEA/D algorithms can obtain good approximations on most of the problems especially on bi-objective problems.
Comparing to generic offspring production operators, the operators based on probabilistic models are computational costly. In the case of MOEA/D-MG, the number of multivariate Gaussian models built in each generation is the same as that of sub-problems. Recall that the computational time is mostly spent on modeling and sampling, however only one trial solution is sampled for each model. Thus the future work to improve the performance of MOEA/D-MG may include (1) sampling more solutions for each model, and (2) reducing the number of models built in each generation.

ACKNOWLEDGMENT

This work is supported by National Basic Research Program of China (No.2011CB707104) and National Science Foundation of China (No.61005050).

REFERENCES