# Adaptive Memetic Computing for Evolutionary Multiobjective Optimization

Vui Ann Shim, Kay Chen Tan, and Huajin Tang

***Abstract*—Inspired by biological evolution, a plethora of algorithms with evolutionary features have been proposed. These algorithms have strengths in certain aspects, thus yielding better optimization performance in a particular problem. However, in a wide range of problems, none of them are superior to one another. Synergetic combination of these algorithms is one of the potential ways to ameliorate their search ability. Based on this idea, this paper proposes an adaptive memetic computing as the synergy of a genetic algorithm, differential evolution, and estimation of distribution algorithm. The ratio of the number of fitter solutions produced by the algorithms in a generation defines their adaptability features in the next generation. Subsequently, a subset of solutions undergoes local search using the evolutionary gradient search algorithm. This memetic technique is then implemented in two prominent frameworks of multiobjective optimization: the domination- and decomposition-based frameworks. The performance of the adaptive memetic algorithms is validated in a wide range of test problems with different characteristics and difficulties.**

***Index Terms*—Differential evolution, estimation of distribution algorithm, evolutionary gradient search, genetic algorithm, memetic computing, multiobjective optimization.**

## I. Introduction

MANY real-world problems involve multiple objectives that are required to be optimized simultaneously. The problems become more difficult to solve when the objectives are conflicting and bounded by constraints. This type of problems is commonly known as multiobjective optimization problems (MOPs). Without loss of generality, minimization problems are considered throughout the paper. Mathematically, an MOP is interpreted as follows:

$$\begin{aligned} &\text{Minimize: } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ &\text{subject to: } \mathbf{g}(\mathbf{x}) \leq 0 \text{ and } \mathbf{h}(\mathbf{x}) = 0 \end{aligned} \tag{1}$$

where $\mathbf{f}(\mathbf{x})$ is the objective vector, $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$, $\mathbb{R}^m$ is the objective space, $m$ is the number of objective functions, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is the decision vector, $\mathbf{x} \in \mathbb{R}^n$, $\mathbb{R}^n$ is the decision space, $n$ is the number of decision variables, $\mathbf{g}$ is the set of inequality constraints, and $\mathbf{h}$ is the set of equality constraints. Due to the conflicting property of the objective function, the optimal solutions consist of a set of tradeoff solutions rather than a single optimal solution. Thus, the definitions of optimality, which have been defined as Pareto set (PS) and Pareto front (PF), is different from single-objective problems. PS is the optimal set of solutions in the decision space when no other solutions are dominating them. The mapped solution set in the objective space is PF.

Over the past two decades, the studies of MOP have been surrounded on two main issues—the algorithm and framework issues. The algorithm issue focuses on the implementation of different optimizers, such as genetic algorithm (GA) [1], differential evolution (DE) [2], and estimation of distribution algorithm (EDA) [3], to explore and exploit the search space. Many studies showed that these algorithms are efficient in solving particular test problems but none of them can be treated as a universal optimizer [4]. On the other hand, the framework issue tackles the problems of how to define the superiority of solutions and how to maintain the multiple tradeoff Pareto optimal solutions. Two prominent frameworks of multiobjective evolutionary algorithms (MOEAs) are domination- and decomposition-based frameworks. In the domination-based MOEAs, the fitness of a solution is assigned according to some principles that define the domination behaviors among the solutions. Then, all objectives are optimized simultaneously [5]. In the decomposition-based framework, an MOP is first transformed into multiple scalar optimization problems (known as sub-problems). All sub-problems are then optimized concurrently. Even though the optimization performance of these frameworks has been extensively studied, their comparison in a wide range of test problems is considerably lacking.

Inspired by the notion of a meme and natural evolution, memetic algorithm (MA) has been introduced as population-based hybrid GAs with local refinement capability [6]. Since then, many variants of MAs have been proposed that involve the synergy or combination of simple agents and memes [4], multiple heuristics or operators [7], multiple population-based global search algorithms [8], or a marriage between a population-based global search and local search algorithms [9], [10]. Besides, instead of merging several existing optimizers or operators, there were other MAs that employed randomized perturbations to the population [11], different stages of exploration which performed periodically [12], and parallel alteration of a solution through different strategies [13]. In [4], MAs are defined as evolutionary algorithms with local

Manuscript received December 23, 2013; revised April 10, 2014 and June 17, 2014; accepted June 17, 2014. Date of publication July 8, 2014; date of current version March 13, 2015. This paper was recommended by Associate Editor H. Ishibuchi.

V. A. Shim and H. Tang are with the Institute for Infocomm Research, Singapore 138632 (e-mail: shimva@i2r.a-star.edu.sg; htang@i2r.a-star.edu.sg).

K. C. Tan is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: eletankc@nus.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2331994

search within the generation cycle while memetic computing approaches involve hybrid structures such as perturbing a single solution. Though distinctions exist among those MAs, all of them stand on the conjecture that synergy among different optimizers, operators, or strategies is a promising way to achieve a common goal through the propagation of memes into a subpopulation of recipients or sharing information within populations.

The aims of this paper are threefold. First, an adaptive memetic computing with the synergy of a GA, DE, and EDA is proposed. The ratio of the number of fitter solutions produced by the algorithms in a particular generation defines their adaptability features in the next generation. Subsequently, a subset of the produced solutions undergoes local search in order to exploit a fitter solution. This synergy implies the communication among different pools of agents such that memes, which are the information or features that can improve the fitness of a chromosome [14], are shared and broadcasted into the whole population. Second, the adaptive memetic computing is implemented in the domination- and decomposition-based frameworks of multiobjective optimization, thus two MAs are devised. Third, the optimization performance of the proposed algorithms are validated and analyzed in 38 benchmark bounded-constrained test instances with different characteristics and difficulties. The preliminary results of this paper have been presented in [15]. A more detailed description, result, analysis, and discussion are provided in this paper.

The proposed MAs lie on two main issues, how to adaptively exchange information according to the peculiarity of the fitness landscapes or problem characteristics and how to choose suitable optimizers such that sharing information among the optimizers is beneficial. For the first issue, if an optimizer has identified a promising search region, more information is drawn from this optimizer as it is closer to that landscape. However, a control mechanism should be introduced to progressively monitor the search such that the balance of information sharing can be governed. For the second issue, we choose optimizers that have vastly different characteristics in producing offspring. EDA considers probability distribution of solutions in the fitness landscape and only uses global probability information throughout the evolutionary process, DE has high exploration capability to generate diverse solution set, GA has a good balance between exploration and exploitation, and EGS only uses local information to perform exploitation. With regards to the above design principles, the contributions of this paper are summarized as follows.

1) We devise an adaptive memetic computing for multiobjective optimization. Though the adaptive principle proposed in this paper has similar concept as AMALGAM [16] and Borg MOEA [17], a progressively control paradigm is missing in both algorithms.
2) We consider multiple global and a local search algorithm in the adaptive memetic computing. Both AMALGAM and Borg MOEA do not involve any local search algorithm. Besides, different optimizers are implemented in the proposed algorithms.
3) We implement and compare the proposed adaptive memetic computing in the domination- and decomposition-based frameworks of multiobjective optimization. AMALGAM is developed in a domination-based framework while Borg MOEA is developed in an indicator-based framework.
4) Our MAs outperform many other state-of-the-art algorithms, including AMALGAM and Borg MOEA, in a wide variety of well-known test problems.

The structure of the paper is as follows. Section II presents the relevant background information. The adaptive MAs are described in Section III. The experimental setup and test problems are depicted in Section IV. Section V presents the simulation results and discussion, and conclusion is drawn in the last section.

## II. Background Information

### A. MOEAs

*1) MOEAs Using Genetic Algorithms:* GA is one of the most popular algorithms used to solve MOPs. NSGA-II [1] is a well-known MOEA that implements GA in the domination-based framework. MOEA/D [18], on the other hand, implements GA in the decomposition-based framework and has become an eminent algorithm due to its superior optimization performance. Both of these algorithms uses simulated binary crossover (SBX) and polynomial mutation operators as their variation operators to explore and exploit the search space. In the SBX operator, the exploration is performed by mating two solutions that are far from each other in the search space. It is also effective in exploiting the promising search regions by generating child solutions that are arbitrary close to the parent solutions. However, the SBX operator suffers several limitations. As pointed out in [19], it may lose diversity when an MOP has a complicated PS and it often generates inferior solutions during an evolutionary process. A detailed description of the SBX can be referred to [20].

*2) MOEAs Using Differential Evolution:* DE has gained increasing interest from the research community due to its simplicity and strong exploration capability. Some examples of DE for solving MOPs are nondominated sorting differential evolution (NSDE) and MOEA/D with DE [19]. For NSDE, Iorio and Li [21] took advantages from the framework of NSGA-II and applied DE as its genetic operator instead of GA. NSDE showed a better performance in rotated MOPs than NSGA-II. Li and Zhang [19] implemented DE in MOEA/D and the algorithm showed superior performance in MOPs with complicated PS. This is because DE is able to generate solutions with huge dissimilarity in their decision vectors, thus producing a set of diverse child solutions. However, DE suffers several limitations such as it has a poor performance in not linearly separable problems [22] and is easily to be trapped in local optima in some problem landscapes [23]. The technical details of DE can be referred to [19].

*3) MOEAs Using Estimation of Distribution Algorithms:* GA and DE employ the concept of genetic variations, such as crossover and mutation, in producing child solutions. EDAs, on the other hand, have a different variation concept that the child solutions are generated by the probability distribution of parent solutions. Some examples of those techniques

are Bayesian network [24] and restricted Boltzmann machine (RBM) [25]. Experimental studies indicated that EDAs are well suited for epistatic and deceptive test problems [24], [26], are robust in noisy environments [27], [28], and are well-performed in high-dimensional MOPs [29]. The downsides of EDAs include the building of the probabilistic model is time consuming [30], they are weak in producing a diverse set of solutions, and are easily trapped in local optima. This is due to the fact that EDAs only use global information, in terms of probability distribution, and do not utilize location information of the known solutions in a reproduction stage [30], [31]. In this paper, only EDAs with RBM (REDA) as its modeling approach is considered. A complete evolutionary process of REDA involves three main steps. First, the parent solution are used as the training data for RBM. It is then trained by the unsupervised learning of contrastive divergence [32], [33] such that the network reaches a certain level of thermal equilibrium. In the second step, a probabilistic model is constructed based on the energy function of the network. Third, child solutions are produced by sampling the constructed probabilistic model. The technical details of REDA can be referred to [31].

*4) Evolutionary Gradient Search Algorithm:* Local search algorithms introduce a small perturbation to existing solutions such that slightly different solutions are obtained. The hybridization between global and local search algorithms has been experimentally shown to provide a better search performance [34], [35]. In this paper, we implement the EGS as a local search algorithm which utilizes the gradient information of the trajectory movement of solutions. The detail of the algorithm is presented in Algorithm 1. The $\varepsilon$ is set to 1.8 as suggested in [36]. The EGS works as follows. An initial solution ($\mathbf{x}^j$) is randomly selected from a population or selection pool. Normal mutation with zero mean and variance of $\sigma^2$ is applied to introduce a small perturbation to $\mathbf{x}^j$ in order to create $L$ local neighbors. The local neighbors are evaluated by calculating their objective functions. The global gradient direction is then computed as illustrated in Step 5 of Algorithm 1. An offspring is subsequently generated based on the global gradient direction and step size. Next, step size and solution are updated. The selected solution is the one with a lower objective value. In this way, the EGS can search for a fitter solution progressively. The above process is repeated, until the stopping criterion is reached, by treating the selected solution as an initial solution.

**Algorithm 1** Pseudo Code of an Evolutionary Gradient Search (EGS) Algorithm

**Begin**
  **1. Input:** Define initial step size $\sigma_0$
  **Do while (Stopping criterion is not met)**
    **For** $j$ = *1:LS* (Solutions to undergo local search)
      **2. Initial solution:** Select a solution $\mathbf{x}^j$ from the selection pool
      **3. Reproduction:** Create $L$ local neighbors $\mathbf{r}^i, i \in \{1, 2, \ldots, L\}$ by perturbing $\mathbf{x}^j$ using normal mutation $N(0, \sigma^2)$
      **4. Evaluation:** Calculate the objective values of $\mathbf{r}^i$, $F(\mathbf{r}^i)$
      **5. Direction:** Estimate the global gradient direction

$$\hat{v} = \frac{\sum_{i=1}^{L} \left[F(\mathbf{r}^i) - F\left(\mathbf{r}^j\right)\right]\left(\mathbf{r}^i - \mathbf{r}^j\right)}{\| \sum_{i=1}^{L} [F(\mathbf{r}^i) - F(\mathbf{r}^j)](\mathbf{r}^i - \mathbf{r}^j) \|}$$

      **6. Offspring generation**

$$\mathbf{y} = \mathbf{x}^j - \sigma_t \hat{v}$$

      **7. Mutation step size update**

$$\sigma_{t+1} = \begin{cases} \sigma_t \varepsilon & \text{if } F(\mathbf{y}) < F(\mathbf{x}^j) \\ \sigma_t / \varepsilon & \text{otherwise} \end{cases}$$

      **8. Solution update:**
      **if** $F(\mathbf{y}) < F(\mathbf{x}^j)$ **then**
      $\mathbf{x}^j = \mathbf{y}$
      **end if**
      **9. Output:** Output $\mathbf{x}^j$
    **End For**
  **End Do**
**End**

### B. Algorithm Frameworks

*1) Domination-Based Framework:* In domination-based MOEAs, the fitness of a solution is determined by the domination principle. This fitness becomes the criterion in natural selection. The final output is a set of tradeoff and nondominated solutions. For preserving the diversity of solutions, techniques such as crowding distance or niche sharing are applied. Some MOEAs under this framework are strength Pareto evolutionary algorithm (SPEA) [37] and NSGA-II. However, this approach suffers several limitations. A large number of nondominated solutions exist in many-objective problems. This leads to a phenomenon called dominance resistance [17], [38], [39], in which it is hard to determine which solutions are promising in producing a fitter offspring. More information on many-objective problems can be referred to the study of NSGA-III [40], HypE [41], PICEA-g [42], and cornet sort approach [43]. Besides, a diversity preservation scheme is necessary to maintain the diversity of solutions. Even through a diverse solution set exists, the distribution of the solutions may not be well maintained [18].

*2) Decomposition-Based Framework:* In decomposition-based MOEAs, an MOP is decomposed into several scalar optimization sub-problems. Subsequently, a tradeoff optimal solution set is determined by optimizing all the sub-problems concurrently. This framework has gained increasing interest from the research community after Zhang and Li [18] devised a multiobjective optimization algorithm based on decomposition (MOEA/D). In MOEA/D, an MOP is decomposed into $N$ scalar optimization sub-problems using weighted sum, Tchebycheff, or boundary intersection approaches. The superiority of solutions is determined by comparing their scalar objective values. Subsequently, new solutions are produced by mating the solution of a sub-problem with its neighboring solutions. In [44], a new MOEA/D with dynamic resource allocation (MOEA/D-DRA) was proposed and tested on CEC09 unconstrained MOPs test problems. In this MOEA/D, different computational efforts are assigned to different sub-problems. This algorithm has been granted the best MOEA in the CEC09 competition for unconstrained multiobjective optimization.

### C. Memetic Algorithm

In this section, only a brief review of MAs in the perspective of MO is presented. In [45], a multiobjective genetic local search algorithm (MOGLS) was introduced. In this MA, the multiple objectives of an MOP are aggregated using a weighted sum approach. A weight vector is randomly generated for each solution that undergoes local search. Another attempt to use the weighted sum approach in performing local search for solving MOPs was suggested in [46]. The weighted sum approach is introduced to NSGA-II, in which the weight vector is not randomly generated, but determined by the information carried on each solution. Besides, the weight vector can also be determined by using a simulated annealing algorithm as described in [47].

In [48], the authors proposed an MA for Pareto archived evolution strategy algorithm (M-PAES). An evolution strategy acts as a local search algorithm and recombination of archived solutions acts as a global search mechanism. Goh *et al.* [36] developed a multiobjective evolutionary gradient search (EGS) algorithm. The population-based approach of EAs is adapted into the EGS algorithm, such that the EGS algorithm can perform a multidirectional search in the search space. Several approaches are used to derive the gradient information of multiple objectives of an MOP, such as a weighted sum approach with random weight vector, a goal programming technique, and a hypervolume indicator-based approach. The detailed coverage of this topics can be referred to [4], [8].

## III. Proposed Algorithms

Motivated by the synergetic effect of multiple EAs, an adaptive memetic computing is suggested to ameliorate the optimization performance of each individual EA. The proposed memetic technique is implemented in the domination- and decomposition-based MOEAs. Thus, two MAs, namely memetic nondominated evolutionary algorithm (mNSEA) and memetic MOEA/D (mMOEA/D), are designed.

The operations of the adaptive memetic computing are as follows. In the initialization stage, each optimizer has an equal probability to produce initial solutions. Then, fitter solutions are selected and stored in an archive. Before child solutions are generated, the reproduction probability of each optimizer is calculated as illustrated in Algorithm 2. It is based on the idea that a higher probability is given to the optimizers that produce a higher number of fitter solutions in the previous generation, and vice versa. Besides, a control parameter called learning rate ($\epsilon$) is introduced to govern the adaptive activity.

Let $\psi$ denotes the solutions in an archive. First, calculate the number of solutions in $\psi$ that are generated by each EA, denoted as $D_g^{EA_i}$, where $i \in \{1, \ldots, M\}$, $M$ is the number of EAs that are involved in the adaptive memetic process. In this paper, three EAs are considered, including a GA, DE, and EDA. Thus, the number of solutions in $\psi$ that are generated by each EA is denoted as $D_g^{EA_1} = D_g^{GA}$, $D_g^{EA_2} = D_g^{DE}$, and $D_g^{EA_3} = D_g^{EDA}$. Afterward, the adaptive proportion rate ($Ar_g^{EA_i}$) at generation $g$ for each EA is calculated as follows:

**Algorithm 2** Pseudo Code of the Adaptive Memetic Computing

1: *%%Given a set of selected solutions that are stored in an archive($\psi$)*
2: **Step 1:** Calculate the number of solutions in $\psi$ that are generated by each EA, denoted as $D_g^{EA_i}$, where $i \in \{1, \ldots, M\}$, $M$ is the number of EAs that are involved in the adaptive memetic process
3: **Step 2:** Calculate the adaptive proportion rate, denoted as $Ar_g^{EA_i}$, of each EA
4: **Step 3:** Check for the lower bound (*l_bound*) of the adaptive proportion rate
5: **For** *i = 1:M*
**if** $Ar_g^{EA_i} < l\_bound$ **then**
$Ar_g^{EA_i} = l\_bound$
**end if**
6: **End For**
7: **Step 4:** Normalize the adaptive proportion rate so that the sum of the adaptive proportion rates is equal to 1.0
8: **For** *i = 1:M*
$Ar_g^{EA_i} = \frac{Ar_g^{EA_i}}{\sum_j^M Ar_g^{EA_j}}$
9: **End For**

$$Ar_g^{EA_i} = Ar_{g-1}^{EA_i} + \epsilon \times Pr_g^{EA_i} \qquad (2)$$
$$Pr_g^{EA_i} = \frac{D_g^{EA_i}}{N}$$

where $Ar_g^{EA_i}$ is the adaptive proportion rate at generation $g$ for $i^{th}$ EA, $\epsilon$ is the learning rate, $Pr_g^{EA_i}$ is the current proportion rate, and $N$ is the archive size or the number of solutions in an archive. The learning rate ($\epsilon > 0$) is introduced to the updating rule to moderate the influence of an optimizer to the whole evolutionary process and it is a scalar parameter to be determined by the user. Next, the adaptive proportion rate is set to a lower bound (*lbound*) value if it is lower than the *lbound* as indicated in Step 3 of Algorithm 2. This is to prevent the stagnation of adaptivity of the MAs during evolutionary processes. Finally, the adaptive proportion rates are normalized such that the summation of the rates is equal to 1.0 (Step 4 of Algorithm 2). After obtaining the proportion rates, a typical evolutionary process continues.

The proposed adaptive method in this paper shares similar adaptive principle as AMALGAM [16], [49] and Borg MOEA [17]. The primarily difference lies in how to determine the final reproduction rate. In AMALGAM, instead of determining reproduction rate, it determines the number of offspring to be generated by each optimizer based on the ratio of the number of offspring generated by the optimizer in the new population. Borg MOEA determines the reproduction rate by calculating the ratio of the number of offspring generated by an algorithm in an $\epsilon$-box dominance archive. Both algorithms determine the final reproduction rate or number of offspring to be generated by each optimizer based on the solutions in an archive or population. The proposed adaptive method, on the

**Algorithm 3** Pseudo-Code of the Adaptive Memetic Nondominated Sorting Evolutionary Algorithm (mNSEA)

```
Begin
  1. Initialization: At generation g = 0, randomly generate N solutions to be the initial population Pop(g)
  2. Evaluation: Evaluate each solution in the population
  Do while (Stopping criterion is not met)
    3. Fitness assignment: Apply the Pareto ranking and crowding distance over the population.
    4. Selection: Select N solutions (binary tournament)
    5. Adaptive: Calculate the adaptive proportion rate
    6. Reproduction: Build a probabilistic model p_g(x)
      For i = 1:N
        Generate a random value between [0, 1] (u)
        if u < Ar_g^GA then
          Generate an offspring using GA operators
        else if u ≥ Ar_g^GA & u < Ar_g^GA + Ar_g^DE then
          Generate an offspring using DE operators
        else
          Sample an offspring from p_g(x) (EDA)
        end if
      End For
    7. Evaluation: Evaluate all child solutions
    8. Archiving: Store the parents and child solutions in an archive. Perform the Pareto ranking and crowding distance over the solutions in the archive
    9. Elitism: Select N solutions with the lowest Pareto rank or highest crowding distance from the archive
    10. Local Search: Generate a random value between [0, 1] (u)
    if u < LS (local search rate) then
      For i = 1:N
        Generate another random value between [0, 1] (u)
        if u < K (% of solutions undergoing local search) then
          Perform EGS algorithm to generate an offspring
        end if
      End For
    end if
    Perform archiving and elitism to the parents and child solution to form the new population. g++
  End Do
End
```

other hand, introduces a control mechanism ($\epsilon$ or learning rate) to progressively monitor the contribution of each optimizer to an evolutionary process. This is to ensure that a balance of information sharing can be governed. This is due to the fact that some optimizers have different convergence rate at different stage of evolutions. Without the learning rate, certain optimizers may promptly dominate the reproduction process, thus ignoring the contributions of other optimizers.

The process flow of the proposed adaptive memetic evolutionary algorithm with nondominated sorting approach (mNSEA) is presented in Algorithm 3. The initial solutions are randomly generated and then evaluated. Based on their objective values, the solutions are sorted into different levels of domination. The solutions in the first level are not dominated by any other solutions while the solutions in the second level are only dominated by the solutions in the first level, and so on. In order to determine the superiority of solutions located on the same level, crowding distance is calculated. A solution is fitter when it is in a higher level and has a larger crowding distance. Based on these criteria, $N$ promising solutions are selected using the binary tournament selection operator. Next, the adaptive proportion rate for each EA is calculated as presented in Algorithm 2. To generate child solutions, a random number is used to determine which EAs are activated to reproduce. If GA is activated, the SBX and polynomial mutation operators are used to create an offspring. If DE is activated, the DE and polynomial mutation operators are used to generate an offspring. Similarly, if EDA is activated, then an offspring is sampled from the constructed probabilistic model. In the archiving state, the parent and child solutions are stored in an archive ($2N$ size) and their levels of domination and crowding distance are determined. In the elitism state, the best $N$ solutions are selected to be a new population. Subsequently, a subset of the population ($K\%$ solutions) is chosen to undergo local search using the EGS algorithm. Note that local search is only activated in certain generations, in which the activation is governed by a local search rate ($LS$). The archiving and elitism stages are also applied to the solutions generated by the EGS algorithm. The similar process repeats until the stopping criterion is reached.

The adaptive memetic computing is also implemented in MOEA/D, and dubbed as mMOEA/D. The complete operation of mMOEA/D is presented in Algorithm 4. In this paper, we apply the MOEA/D proposed by Li and Zhang [19]. The Tchebycheff approach is used to decompose an MOP into $T$ scalar sub-problems. Its weight vectors $(\boldsymbol{\lambda}^i, \ldots, \boldsymbol{\lambda}^N)$ are uniformly generated before the evolutionary process starts. It is to be noted that $T = N$, which means that each sub-problem is represented by a solution in the population. The neighboring solutions of a sub-problem are determined by the Euclidean distance among the weight vectors. Each sub-problem is assigned $Q$ neighboring solutions, denoted as $B(i) = \{i_1, \ldots, i_Q\}, i \in [1, N]$. A set of initial solutions is randomly generated and evaluated. Note that the fitness of a solution is set to be the value of the scalar function. The initial reference point of the Tchebycheff approach ($\mathbf{z}^*$) is initialized to be the lowest objective values of the solutions in the population. Next, the evolutionary process starts.

Similar to mNSEA, a random number is used to determine which EAs are activated to reproduce. If GA is activated, a child solution is generated by mating two randomly selected neighboring solutions of solution $i$ using the SBX operator and then followed by the perturbation of polynomial mutation operator. If DE is activated, three solutions are involved in the reproduction. The solution $i$ is modified through the mating with another two randomly selected neighboring solutions of solution $i$. The polynomial mutation with a probability of $p_m$ is applied to the resultant child solution. If EDA is activated, a child solution is sampled from the probabilistic models. Two probabilistic models are constructed by the RBM. The first model constructs the probabilistic model from all solutions while the second model only builds the probability distribution of the neighboring solutions of solution $i$. Next, the newly generated solutions are evaluated, and reference point ($\mathbf{z}^*$) is updated. If the child solution $i$ is fitter than the parent

**Algorithm 4** Pseudo-Code of the Adaptive Memetic MOEA/D (mMOEA/D)

**Begin**
**1. Initialization:**
a. Generate a set of uniformly distributed weight vectors $(\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^N)$
b. Calculate the Euclidean distance between the weight vectors. Determine $Q$ neighboring solutions ($B(i) = \{i_1, \ldots, i_Q\}, i \in [1, N]$) for each weight vectors according to the short Euclidean distance
c. At generation $g = 0$, randomly generate $N$ solutions to be the initial population $Pop(g)$
d. Initialize reference point of the Tchebycheff approach ($\mathbf{z}^*$) by setting the value of $\mathbf{z}^*$ to be the lowest objective values of the solutions
**Do while (Stopping criterion is not met)**
Build a probabilistic model of the whole population $p_g(x)$
**For** $i$ = *1:N*
**2. Reproduction:** Generate a random value between [0, 1] ($u$)
**if** $u < Ar_g^{GA}$ **then**
Randomly select two solutions from $B(i)$, and then generate an offspring using GA operators
**else if** $u \geq Ar_g^{GA} \& u < Ar_g^{GA} + Ar_g^{DE}$ **then**
Randomly select three solutions from $B(i)$, and then generate an offspring using DE operators
**else**
Build another probabilistic model $p_g^{'}(\mathbf{x})$ which only considers the neighboring solutions. Generate a random value between [0, 1] ($u$)
**For** $j = 1{:}n$ (number of decision variables)
**if** $u < 0.5$ **then** Sample from $p_g(\mathbf{x})$
**else** Sample from $p_g^{'}$ **end if**
**End For**
**end if**
**3. Evaluation:** Evaluate the generated offspring $\mathbf{y}$ to obtain the corresponding objective values $\mathbf{f}(\mathbf{y})$
**4. Update of $\mathbf{z}^*$: For** $j = 1, \ldots, m$, **if** $z_j^* > f_j(\mathbf{y})$ **then** set $z_j^* = f_j(\mathbf{y})$
**5. Fitness assignment:** Assign fitness to each solution ($g^{te}$) using Tchebycheff method
**6. Update solution: For** $j \in B(i)$, **if** $g^{te}(\mathbf{y}|\boldsymbol{\lambda}^j, \mathbf{z}^*) \leq g^{te}(\mathbf{x}^j|\boldsymbol{\lambda}^j, \mathbf{z}^*)$, **then** set $\mathbf{x}^j = \mathbf{y}$ and $FV^j = \mathbf{f}(\mathbf{y})$
**End For**
**7. Local search:** Perform the EGS algorithm if it is activated (similar to hNSEA). Then, apply Steps 4-6 to update the reference point and the neighboring solutions. $g++$
**End Do**
**End**

solution $i$ and its neighboring solutions, the parent solution $i$ and the neighboring solutions are discarded and replaced by the child solutions $i$. Next, if local search is activated, the EGS algorithm is used to produce child solutions as illustrated in Algorithm 1. The similar process repeats until the stopping criterion is reached.

## IV. Problem Description and Implementation

The optimization performance of the proposed algorithms (mNSEA and mMOEA/D) was validated in 38 benchmark test instances with different characteristics and difficulties. The test problems include five ZDT problems [50], seven DTLZ problems [51] with three objectives and five objectives respectively, ten UF problems [52], and nine WFG problems [53].

Besides, their performance was compared with other eight MOEAs, including NSGA-II with SBX operator (NSGA-II-SBX) [1], NSDE [21], MOEA/D with SBX operator (MOEA/D-SBX) [18], MOEA/D with DE (MOEA/D-DE) [19], NSREDA [25], MOEA/D with REDA (MOEA/D-REDA), AMALGAM [16], and Borg MOEA [17]. The algorithms start with NS are the domination-based MOEAs, and those start with MOEA/D are the decomposition-based MOEAs. The SBX operator was used in GA, DE/rand/1/bin operator was used in DE, and RBM was used in EDA. We also constructed REDA in the decomposition-based framework (MOEA/D-REDA) for a thorough comparison. Note that we implemented the adaptive updating rule proposed in [16] into our domination-based algorithm and the algorithm is dubbed as AMALGAM (a term used in [16]). In other words, the difference between mNSEA and AMALGAM is the implementation of different adaptive updating rule, in which a control parameter is introduced in the updating rule of mNSEA while this feature is missing in AMALGAM. The source code of Borg MOEA was obtained from the authors of [17]. It is to be noted that no local search is involved in AMALGAM and Borg MOEA.

The optimization performance of the algorithms was accessed using inverted generational distance (IGD). IGD is a unary indicator which measures the distance of each solution in the optimal PF to the obtained PF. In this indicator, both convergence and diversity are taken into consideration. A lower value of IGD implies that an algorithm has better optimization performance. Based on IGD values, the ranking of ten algorithms was performed. The details of the parameter settings were summarized in Table I.

TABLE I
Parameter Settings for Experiments

| Parameter | Setting |
|---|---|
| Population size, $N$ | 100, 300, 500 (for 2, 3, and 5 objectives) |
| Stopping criterion | $500 \times N$ fitness evaluations |
| Number of runs | 10 |
| $lbound$ in the adaptive feature | 0.1 |
| $\epsilon$ in the adaptive feature | 0.1 |
| Number of hidden units in REDA | 5 |
| Number of training epochs in REDA | 2 |
| Distribution index ($\varphi$) | 20 |
| Mutation rate ($p_m$) | 1/(Number of variables) |
| DE and crossover rate of the SBX | 0.9 |
| Local search rate ($LS$) | 0.5 |
| $K$ in the EGS algorithm | 10% of the population size |
| $L$ in the EGS algorithm | 4 |
| $Q$ in MOEA/D | 20 |

## V. Results and Discussions

### A. Comparison Results

The performances measured in IGD for all test problems are presented in Tables II and III. The number of decision

TABLE II
RESULTS IN TERMS OF IGD MEASUREMENT FOR ZDT, DTLZ, UF, AND WFG TEST PROBLEMS

| Algorithm | Test Instance $(m, n)$ | | | | | |
|---|---|---|---|---|---|---|
| | ZDT1(2,300) | ZDT2(2,300) | ZDT3(2,300) | ZDT4(2,100) | ZDT6(2,100) | DTLZ1(3,12) |
| NSGA-II-SBX | 0.1979±0.0296(7) | 0.3791±0.0564(7) | 0.1585±0.0208(5) | 25.320±2.0645(5) | 0.9203±0.0558(5) | 0.0146±0.0004(2) |
| MOEA/D-SBX | 0.2997±0.0319(8) | 0.4747±0.1405(8) | 0.3141±0.0399(8) | 29.053±3.6616(7) | 0.6175±0.0391(3) | 0.0173±0.0001(5) |
| NSDE | 1.2686±0.1235(10) | 2.5681±0.2241(10) | 0.8484±0.1112(9) | 27.687±6.2070(6) | 5.5331±0.2267(10) | 1.2042±1.8231(8) |
| MOEA/D-DE | 1.0675±0.0353(9) | 1.9065±0.1697(9) | 0.9205±0.0442(10) | 35.388±8.4534(9) | 3.7106±0.2538(9) | 0.1828±0.4989(6) |
| NSREDA | 0.1497±0.0047(5) | 0.2715±0.0084(4) | 0.1696±0.0040(6) | 18.217±1.5058(4) | 3.1699±0.0920(8) | 11.006±1.8385(10) |
| MOEA/D-REDA | 0.1895±0.0077(6) | 0.3521±0.0189(5) | 0.2016±0.0054(7) | 16.253±1.9515(3) | 3.0413±0.0077(7) | 8.4998±0.9996(9) |
| mNSEA | **0.0148±0.0039(1)** | **0.0161±0.0008(1)** | **0.0126±0.0005(1)** | **7.3030±4.4287(1)** | **0.0103±0.0170(1)** | 0.0169±0.0012(3) |
| mMOEA/D | 0.0475±0.0062(2) | 0.0683±0.0097(2) | 0.0726±0.0088(3) | 36.624±4.6961(10) | 0.6760±0.0381(4) | 0.0172±0.0001(4) |
| AMALGAM | 0.0617±0.0028(3) | 0.0941±0.0106(3) | 0.0709±0.0040(2) | 31.834±4.6024(8) | 1.4446±0.0611(6) | 0.5397±0.5004(7) |
| Borg MOEA | 0.1411±0.0287(4) | 0.3536±0.0567(6) | 0.1368±0.0038(4) | 13.875±2.3774(2) | 0.2842±0.0324(2) | **0.0085±0.0007(1)** |
| Algorithm | DTLZ2(3,120) | DTLZ3(3,12) | DTLZ4(3,120) | DTLZ5(3,120) | DTLZ6(3,120) | DTLZ7(3,120) |
| NSGA-II-SBX | 0.0458±0.0025(3) | 0.0381±0.0025(4) | **0.0567±0.0020(1)** | 0.0229±0.0008(2) | 48.058±1.0496(8) | 0.1008±0.0039(3) |
| MOEA/D-SBX | 0.0327±0.0007(2) | 0.0329±0.0004(2) | 0.2183±0.1878(3) | 0.0240±0.0001(4) | 21.187±1.0420(5) | 0.1742±0.0003(6) |
| NSDE | 1.9567±0.2738(10) | 4.0248±3.1192(8) | 11.860±0.9960(9) | 0.7055±0.1660(10) | 57.367±3.6175(9) | 3.0625±0.5678(10) |
| MOEA/D-DE | 0.0659±0.0035(5) | 0.0561±0.0591(6) | 12.042±0.7354(10) | 0.0274±0.0014(5) | 3.1601±1.6770(4) | 0.4761±0.0701(9) |
| NSREDA | 0.3623±0.0638(9) | 40.970±7.9440(10) | 1.6576±0.3273(8) | 0.2927±0.0691(9) | 71.850±4.3401(10) | 0.4095±0.0254(8) |
| MOEA/D-REDA | 0.0776±0.0072(6) | 28.875±7.5419(9) | 1.0396±0.5263(7) | 0.0377±0.0068(6) | 0.0246±0.0001(2) | 0.1679±0.0037(5) |
| mNSEA | 0.0593±0.0020(4) | 0.0456±0.0028(5) | 0.2010±0.1280(2) | **0.0225±0.0003(1)** | **0.0235±0.0003(1)** | 0.0974±0.0044(2) |
| mMOEA/D | **0.0320±0.0008(1)** | 0.0331±0.0007(3) | 0.2680±0.3012(4) | 0.0239±0.0001(3) | 27.410±4.8198(6) | 0.1743±0.0006(7) |
| AMALGAM | 0.1426±0.0122(7) | 0.4120±0.6045(7) | 0.3299±0.1280(5) | 0.0487±0.0037(7) | 41.544±2.3210(7) | 0.1136±0.0054(4) |
| Borg MOEA | 0.2141±0.0317(8) | **0.0191±0.0096(1)** | 0.3847±0.0660(6) | 0.0561±0.0090(8) | 0.0299±0.0000(3) | **0.0955±0.0034(1)** |
| Algorithm | UF1(2,30) | UF2(2,30) | UF3(2,30) | UF4(2,30) | UF5(2,30) | UF6(2,30) |
| NSGA-II-SBX | 0.1201±0.0244(8) | 0.0479±0.0103(6) | 0.2372±0.0404(6) | 0.0538±0.0022(5) | 0.3087±0.1100(2) | 0.1462±0.0464(7) |
| MOEA/D-SBX | 0.1258±0.0498(10) | 0.0576±0.0297(8) | 0.3094±0.0533(9) | 0.0566±0.0047(6) | 0.4436±0.1030(5) | 0.1744±0.0549(10) |
| NSDE | 0.0522±0.0128(3) | 0.0450±0.0056(5) | 0.1385±0.0318(3) | 0.0730±0.0078(7) | 0.8802±0.1721(10) | 0.0442±0.0088(2) |
| MOEA/D-DE | 0.0489±0.0290(2) | **0.0342±0.0237(1)** | **0.0745±0.0378(1)** | 0.0824±0.0079(8) | 0.6726±0.1360(9) | 0.0466±0.0291(3) |
| NSREDA | 0.1251±0.0395(9) | 0.1023±0.0077(10) | 0.3888±0.0766(10) | 0.1347±0.0105(10) | 0.5600±0.1262(8) | 0.1719±0.0383(9) |
| MOEA/D-REDA | 0.1181±0.0305(7) | 0.0654±0.0075(9) | 0.3088±0.0368(8) | 0.0910±0.0073(9) | 0.4930±0.1314(7) | 0.1629±0.0666(8) |
| mNSEA | 0.0549±0.0078(4) | 0.0383±0.0042(2) | 0.1292±0.0347(2) | 0.0507±0.0030(2) | 0.3163±0.0854(3) | **0.0437±0.0098(1)** |
| mMOEA/D | **0.0396±0.0113(1)** | 0.0410±0.0217(3) | 0.1812±0.0699(4) | 0.0511±0.0030(3) | 0.4871±0.1233(6) | 0.1235±0.1592(5) |
| AMALGAM | 0.0585±0.0186(5) | 0.0484±0.0029(7) | 0.2258±0.0210(5) | 0.0531±0.0017(4) | 0.4430±0.2120(4) | 0.0504±0.0333(4) |
| Borg MOEA | 0.1072±0.0198(6) | 0.0423±0.0053(4) | 0.2863±0.0624(7) | **0.0449±0.0013(1)** | **0.2234±0.0462(1)** | 0.1360±0.0655(6) |
| Algorithm | UF7(2,30) | UF8(3,30) | UF9(3,30) | UF10(3,30) | WFG1(2,30) | WFG2(2,30) |
| NSGA-II-SBX | 0.1682±0.1338(7) | 0.2203±0.0047(8) | 0.1710±0.0423(6) | 0.3274±0.0596(4) | 1.3888±0.0891(10) | 0.1017±0.0652(8) |
| MOEA/D-SBX | 0.3222±0.1379(9) | 0.1607±0.0379(5) | 0.1211±0.0566(2) | 0.3257±0.1810(3) | 1.1816±0.1144(6) | 0.1567±0.0374(10) |
| NSDE | 0.0329±0.0090(4) | 0.1478±0.0143(3) | 0.1822±0.0672(7) | 2.4853±0.2086(10) | 1.2670±0.0118(9) | 0.0306±0.0100(3) |
| MOEA/D-DE | 0.0235±0.0063(2) | **0.0937±0.0082(1)** | **0.1058±0.0485(1)** | 0.6108±0.0706(9) | 1.2309±0.0033(8) | 0.0500±0.0064(5) |
| NSREDA | 0.3386±0.1624(10) | 0.2971±0.0592(9) | 0.4111±0.0418(10) | 0.5053±0.0694(7) | 1.1739±0.0057(5) | 0.0983±0.0298(7) |
| MOEA/D-REDA | 0.2988±0.1439(8) | 0.1937±0.0123(6) | 0.3661±0.0515(9) | 0.4880±0.0780(6) | 1.1832±0.0102(7) | 0.1377±0.0572(9) |
| mNSEA | 0.0249±0.0075(3) | 0.1994±0.0363(7) | 0.1444±0.0575(5) | 0.2629±0.0540(2) | 1.0036±0.0080(3) | 0.0270±0.0027(2) |
| mMOEA/D | **0.0205±0.0063(1)** | 0.1143±0.0103(2) | 0.1326±0.0503(3) | 0.3870±0.1280(5) | **0.9362±0.0096(1)** | 0.0970±0.0626(6) |
| AMALGAM | 0.0548±0.0716(5) | 0.1566±0.0366(4) | 0.1398±0.0466(4) | **0.2465±0.0538(1)** | 1.0976±0.0065(4) | **0.0246±0.0011(1)** |
| Borg MOEA | 0.1156±0.1216(6) | 0.4270±0.0588(10) | 0.2664±0.0290(8) | 0.5905±0.1037(8) | 0.9535±0.0056(2) | 0.0370±0.0034(4) |
| Algorithm | WFG3(2,30) | WFG4(2,30) | WFG5(2,30) | WFG6(2,30) | WFG7(2,30) | WFG8(2,30) |
| NSGA-II-SBX | 0.0250±0.0014(3) | 0.0184±0.0008(3) | 0.0671±0.0010(3) | 0.0481±0.0041(4) | 0.0172±0.0007(3) | 0.0803±0.0038(3) |
| MOEA/D-SBX | 0.0299±0.0033(6) | **0.0155±0.0005(1)** | **0.0665±0.0007(1)** | 0.0447±0.0070(3) | **0.0141±0.0001(1)** | 0.0766±0.0054(2) |
| NSDE | 0.0245±0.0011(2) | 0.0994±0.0036(10) | 0.0733±0.0011(7) | 0.0819±0.0204(10) | 0.0332±0.0018(8) | 0.1272±0.0121(8) |
| MOEA/D-DE | 0.0292±0.0026(5) | 0.0791±0.0103(9) | 0.0673±0.0002(4) | 0.0818±0.0182(9) | 0.0180±0.0008(4) | 0.1119±0.0112(6) |
| NSREDA | 0.0709±0.0197(9) | 0.0325±0.0038(7) | 0.0771±0.0014(10) | 0.0611±0.0059(6) | 0.0421±0.0081(9) | 0.1393±0.0106(10) |
| MOEA/D-REDA | 0.0959±0.0247(10) | 0.0506±0.0058(8) | 0.0756±0.0014(9) | 0.0693±0.0125(7) | 0.0457±0.0215(10) | 0.1261±0.0104(7) |
| mNSEA | 0.0260±0.0010(4) | 0.0219±0.0012(5) | 0.0700±0.0010(6) | 0.0815±0.0162(8) | 0.0189±0.0009(5) | 0.0856±0.0029(4) |
| mMOEA/D | 0.0320±0.0012(7) | 0.0164±0.0007(2) | 0.0668±0.0001(2) | 0.0362±0.0090(2) | **0.0141±0.0001(1)** | **0.0741±0.0023(1)** |
| AMALGAM | **0.0232±0.0007(1)** | 0.0282±0.0019(6) | 0.0747±0.0011(8) | 0.0585±0.0041(5) | 0.0229±0.0007(7) | 0.0984±0.0044(5) |
| Borg MOEA | 0.0598±0.0169(8) | 0.0205±0.0012(4) | 0.0696±0.0009(5) | **0.0314±0.0031(1)** | 0.0193±0.0012(6) | 0.1273±0.0102(9) |
| | WFG9(2,30) | DTLZ1(5,12) | DTLZ2(5,120) | DTLZ3(5,12) | DTLZ4(5,120) | DTLZ5(5,120) |
| NSGA-II-SBX | 0.0200±0.0019(3) | 22.503±13.459(7) | 1.5922±0.0627(6) | 75.415±18.419(7) | 6.0306±0.8802(4) | 4.5888±0.5141(7) |
| MOEA/D-SBX | 0.0177±0.0013(2) | 0.2176±0.0025(3) | 1.0260±0.0501(3) | 1.0020±0.0315(4) | **1.1211±0.1238(1)** | 0.9331±0.0027(2) |
| NSDE | 0.0335±0.0008(4) | 62.370±21.402(8) | 3.9636±3.1544(7) | 217.49±30.572(9) | 21.193±0.6283(9) | 1.3406±0.6291(6) |
| MOEA/D-DE | 0.0348±0.0193(5) | 2600.9±162.91(10) | 1.0535±0.0044(5) | 960.33±373.29(10) | 9.2750±0.6970(5) | 0.9375±0.0035(4) |
| NSREDA | 0.0551±0.0158(9) | 4.9827±8.1962(5) | 10.4467±0.8754(10) | 12.157±5.7649(5) | 25.163±0.6490(10) | 9.1865±0.5106(10) |
| MOEA/D-REDA | 0.0384±0.0193(6) | 3.7365±1.5907(4) | 0.9940±0.0449(2) | 17.991±4.4488(6) | 4.5550±1.0588(3) | 0.9369±0.0030(3) |
| mNSEA | 0.0432±0.0097(7) | 12.359±5.1947(6) | 6.6723±1.2337(8) | 0.9197±0.1300(2) | 17.972±1.1783(7) | 7.0585±0.6700(8) |
| mMOEA/D | **0.0165±0.0006(1)** | 0.2145±0.0002(2) | 1.0302±0.0479(4) | 0.9335±0.0050(3) | 1.7985±0.3771(2) | **0.9307±0.0000(1)** |
| AMALGAM | 0.0670±0.0120(10) | 98.956±14.970(9) | 9.3316±0.6574(9) | 215.18±28.420(8) | 18.393±0.3715(8) | 7.8652±0.4387(9) |
| Borg MOEA | 0.0516±0.0071(8) | **0.0180±0.0023(1)** | **0.3317±0.0489(1)** | **0.0636±0.0136(1)** | 12.629±0.6026(6) | 0.9478±0.1761(5) |

variables ($n$) and objective functions ($m$) of the test problems are indicated by the parentheses next to the test problems. The results (mean±standard deviation) are the averaging IGD of ten independent runs with each run refers to different initialization. The numbers (scores) inside the parentheses next to the IGD values refer to the rank of the algorithms in that test problem. All rankings performed in this section (refer to Table IV) are based on the scores inside the parentheses.

TABLE III
RESULTS IN TERMS OF IGD MEASUREMENT FOR DTLZ6-DTLZ7 (FIVE OBJECTIVES) TEST PROBLEMS

| Algorithm | Test Instance $(m, n)$ | |
|---|---|---|
| | DTLZ6(5,120) | DTLZ7(5,120) |
| NSGA-II-SBX | 96.699±0.7107(10) | 4.8243±0.1021(7) |
| MOEA/D-SBX | 15.582±0.4147(5) | 4.2058±0.0095(5) |
| NSDE | 80.677±7.0645(8) | 9.0199±0.9137(9) |
| MOEA/D-DE | 10.334±2.4412(4) | 11.986±0.4779(10) |
| NSREDA | 20.249±6.3474(6) | 3.6190±0.0245(2) |
| MOEA/D-REDA | 0.9297±0.0013(2) | 4.3186±0.1403(6) |
| mNSEA | **0.9008±0.0000(1)** | 4.1183±0.2459(4) |
| mMOEA/D | 0.9307±0.0000(3) | 6.1212±0.6330(8) |
| AMALGAM | 94.500±2.3486(9) | 3.9297±0.0503(3) |
| Borg MOEA | 56.742±1.9361(7) | **0.4528±0.0192(1)** |

TABLE IV
RANKING OF THE ALGORITHMS IN VARIOUS TEST PROBLEMS

| Rank | Test Problem(Score) | | |
|---|---|---|---|
| | ZDT | DTLZ-3 | UF |
| 1 | mNSEA(5) | mNSEA(18) | mNSEA(31) |
| 2 | Borg MOEA(18) | NSGA-II-SBX(23) | mMOEA/D(33) |
| 3 | mMOEA/D(21) | MOEA/D-SBX(27) | MOEA/D-DE(37) |
| 4 | AMALGAM(22) | mMOEA/D(28) | AMALGAM(43) |
| 5 | NSREDA(27) | Borg MOEA(28) | NSDE(54) |
| 6 | MOEA/D-REDA(28) | AMALGAM(44) | Borg MOEA(57) |
| 7 | NSGA-II-SBX(29) | MOEA/D-REDA(44) | NSGA-II-SBX(59) |
| 8 | MOEA/D-SBX(34) | MOEA/D-DE(45) | MOEA/D-SBX(67) |
| 9 | NSDE(45) | NSREDA(64) | MOEA/D-REDA(77) |
| 10 | MOEA/D-DE(46) | NSDE(64) | NSREDA(92) |
| Rank | WFG | DTLZ-5 | Overall |
| 1 | mMOEA/D(23) | Borg MOEA(22) | mMOEA/D(128) |
| 2 | MOEA/D-SBX(32) | mMOEA/D(23) | mNSEA(134) |
| 3 | NSGA-II-SBX(40) | MOEA/D-SBX(23) | Borg MOEA(172) |
| 4 | mNSEA(44) | MOEA/D-REDA(26) | MOEA/D-SBX(183) |
| 5 | Borg MOEA(47) | mNSEA(36) | NSGA-II-SBX(199) |
| 6 | AMALGAM(47) | NSREDA(48) | AMALGAM(211) |
| 7 | MOEA/D-DE(55) | NSGA-II-SBX(48) | MOEA/D-DE(231) |
| 8 | NSDE(61) | MOEA/D-DE(48) | MOEA/D-REDA(248) |
| 9 | NSREDA(72) | AMALGAM(55) | NSDE(280) |
| 10 | MOEA/D-REDA(73) | NSDE(56) | NSREDA(303) |

ZDT problems, which possess two objective functions and a scalable number of decision variables, are a set of simple MOPs. These problems can be easily solved by all algorithms. In order to increase its difficulty, we enlarged the search space by increasing the number of decision variables ten times greater than its original setting. The results show that mNSEA has a superior performance compared to other algorithms. The ranking is then followed by Borg MOEA, mMOEA/D, and AMALGAM. In other words, all MAs have better optimization performance than individual optimizers. In ZDT4, all algorithms fail to approach PF (indicated by large IGD values). This is due to the fact that ZDT4 is an extremely multimodal problem which consists of many local optima. Comparing the performance of different EAs, EDA has a better result than GA and DE. Comparing between the two frameworks, the domination-based algorithms show a better result than the decomposition-based algorithms. The overall ranking of the algorithms in ZDT problems is mNSEA, Borg MOEA, mMOEA/D, AMALGAM, NSREDA, MOEA/D-REDA, NSGA-II-SBX, MOEA/D-SBX, NSDE, and MOEA/D-DE.

DTLZ problems are well-known for their scalability in both decision and objective spaces. To increase the decision space, we set the number of decision variables to 120 except DTLZ1 and DTLZ3 (12 decision variables) which are highly multimodal test problems. In DTLZ problems with three objective functions, Borg MOEA has the best IGD performance in three test problems while mNSEA has the best performance in two test problems. However, the overall ranking suggests that mNSEA has the best rank while Borg MOEA is only placed at fifth rank (Table IV). This is because Borg MOEA has a lower rank in another three test problems while mNSEA performs well on other test problems. It is also observed that GA with the SBX operator is well-performed in multimodality problems (DTLZ1, and DTLZ3). The performance of EDA and DE are inferior in these test problems. This result is identical to the observation in [25] which explains that EDA, which only uses global probability information, is susceptible to be trapped into local optima. DTLZ4 has a bias landscape when mapping from the decision space to the objective space. In this test problem, NSGA-II-SBX and mNSEA obtain a reasonable result while algorithms with DE operator have a worse performance. No clear indication about the superiority of both frameworks in these test problems. The overall ranking is mNSEA, NSGA-II-SBX, MOEA/D-SBX, mMOEA/D, Borg MOEA, AMALGAM, MOEA/D-REDA, MOEA/D-DE, NSREDA, and NSDE.

In many-objective problems, the assignation of fitness to each solution become more complex. This challenges an optimizer in differentiating the superiority of solutions. The DTLZ with five objective functions belongs to this class of problems. Without performing dimension reduction to the objective function or modifying the selection operator, we directly implemented the proposed algorithms to deal with these problems. It is observed that Borg MOEA achieves the best IGD results in four test problems and is placed at the top rank among all algorithms. The employment of $\epsilon$-box dominance archive in Borg MOEA allows the algorithm to maintain the best solutions in terms of convergence and diversity based on $\epsilon$-dominance criterion. This criterion is efficient to determine the superiority of solutions even in many-objective problems. It is clearly observed that algorithms with the decomposition-based frameworks outperform algorithms with the domination-based framework. This is due to the fact that the superiority of a solution in the decomposition-based framework can be easily identified using the value of the aggregated or scalar sub-problems. In the domination-based framework, the domination behavior of the solutions is hardly been determined in these problems. The results also indicated that EDA and GA has a better performance than DE. The overall ranking is Borg MOEA, mMOEA/D, MOEA/D-SBX, MOEA/D-REDA, mNSEA, NSREDA, NSGA-II-SBX, MOEA/D-DE, AMALGAM, and NSDE.

UF problems were used in the CEC 2009 competition and well-known for their complicated shapes of PS. The first seven problems have two objective functions while the rest of the problems possess three objective functions. MOEA/D-DE has the best performance in four UF test problems, followed by Borg MOEA and mMOEA/D in two test problems, respectively. However, mNSEA is placed at the top rank as it has a

consistent performance in all UF test problems. The algorithms with EDA show inferior performance in most of these problems. This may be attributed to the fact that EDAs have a weak diversification [30] since only global information is used and they do not leverage the benefits of location information. It is also reported in [19] that the ability to produce diverse solutions is critical to address UF problems. Both domination- and decomposition-based algorithms show competitive performance. The overall ranking is mNSEA, mMOEA/D, MOEA/D-DE, AMALGAM, NSDE, Borg MOEA, NSGA-II-SBX, MOEA/D-SBX, MOEA/D-REDA, and NSREDA.

WFG problems are scalable problems in both objective and decision spaces. The problems involve multiple transformations such that various properties, such as deceptive, multimodal, bias, etc., are introduced. In this implementation, 30 decision variables (28 distance-related parameters and two position-related parameters) and two objective functions were applied. mMOEA/D shows the best performance in four WFG problems and has the best rank among all algorithms. Besides, algorithms with GA generally outperform algorithms with DE and EDA. Comparing both frameworks, the decomposition-based algorithms slightly outperform the domination-based algorithms. The overall ranking is mMOEA/D, MOEA/D-SBX, NSGA-II-SBX, mNSEA, Borg MOEA, AMALGAM, MOEA/D-DE, NSDE, NSREDA, and MOEA/D-REDA.

Table IV shows that, overall, mMOEA/D has the best rank, followed by mNSEA and Borg MOEA. These results indicate that the proposed algorithms leverage the benefits of each optimizer in an adaptive memetic manner in most of the test problems. However, the MAs fail in certain test problems such as ZDT4, in which they have the worst IGD results. Comparing each individual optimizer, GA has the best performance in DTLZ (three and five objective functions) and WFG problems. It has medium performance in ZDT and UF problems. For DE, its performance is superior in UF problems, medium in WFG problems, and inferior in ZDT and DTLZ problems with three and five objective functions. For EDA, its performance is superior in ZDT problems, medium in DTLZ problems with three and five objective functions, and inferior in UF and WFG problems. Comparing between the MOEAs frameworks, the domination-based MOEAs are superior to the decomposition-based algorithms in ZDT problems, inferior in DTLZ problems with five objective functions, and comparable in the other test problems. The overall ranking of the algorithms in all the test problems is mMOEA/D, mNSEA, Borg MOEA, MOEA/D-SBX, NSGA-II-SBX, AMALGAM, MOEA/D-DE, MOEA/D-REDA, NSDE, and NSREDA.

Several observations can be made from the above results. First, the adaptive updating rule in mNSEA has superior performance compared to updating rule in AMALGAM. The control parameter introduced in mNSEA moderates the influence of an optimizer to the whole evolutionary process such that every optimizer has a longer life span to contribute during the evolutionary cycle. Second, we infer that indicator-based MOEAs (e.g., Borg MOEA) is more effective than decomposition-based MOEAs in differentiating the superiority of solutions in many-objective problems while domination-based MOEAs are less effective than the former MOEAs.

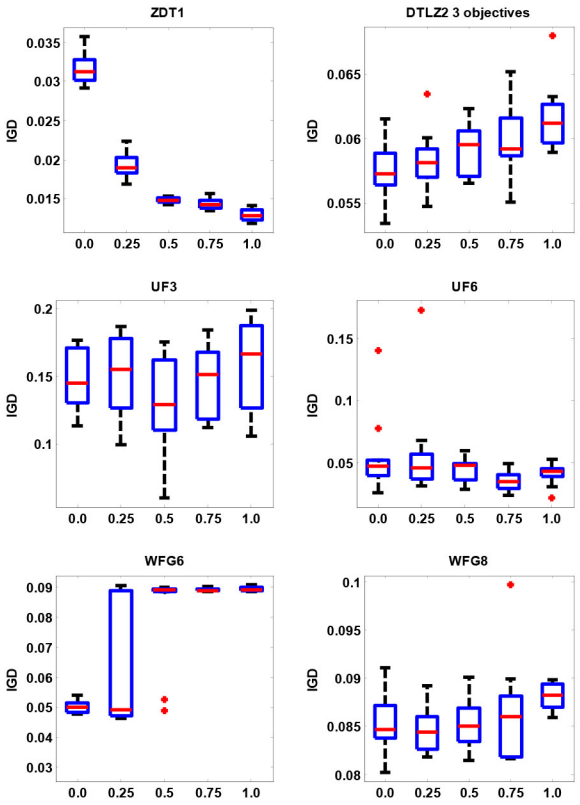


Fig. 1. Effects of local search rate ($x$-axis) on optimization performance.

Third, the proposed adaptive memetic computing is a generic approach which can be implemented in any framework of MOEAs. Lastly, it is to be noted that any optimizer can be integrated into the adaptive memetic computing. However, a better performance is expected when the optimizers show vastly different search characteristics. As an example, GA has a good balance between exploration and exploitation, DE has a strong exploration capability, and EDA has a mechanism to trace the distribution of promising solutions.

### B. Effects of Local Search on Optimization Performance

Fig. 1 shows the effects of different local search rate on optimization performance. Six cases are shown in order to highlight different observations. Overall, the performance of mNSEA is better in most of the test problems when local search is activated (local search rate, $LS = 0$ means local search is deactivated). The box-plot for ZDT1 shows this observation. However, it may also happen that mNSEA has a better performance when $LS = 0$ as shown in DTLZ2 and WFG6. In UF3, $LS = 0.5$ gives the best performance compared to other settings. An interesting observation can be seen in UF6 that the activation of local search reduces the number of outliers. Fig. 2 shows the effects of the number of solutions which undergoes local search on optimization performance. In ZDT1, 25% of solutions in the population undergoing local search give the best performance compared to other settings. In DTLZ2, the performance deteriorates when a higher number of solutions is exposed to local search. The figure of WFG2 shows the reverse observation as DTLZ2, in which the performance is better

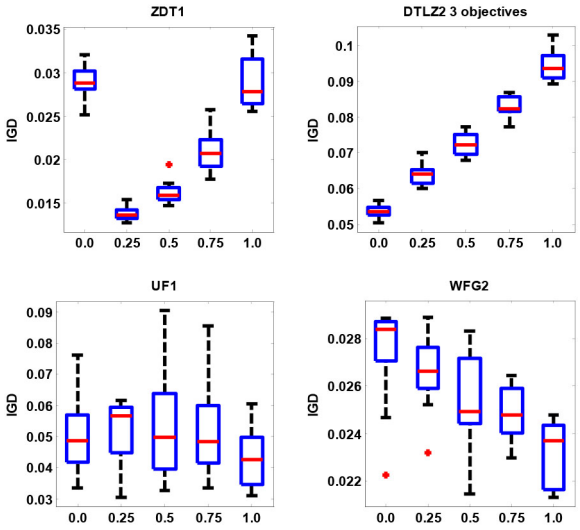


Fig. 2. Effects of the percentage of local search ($x$-axis) on optimization performance.

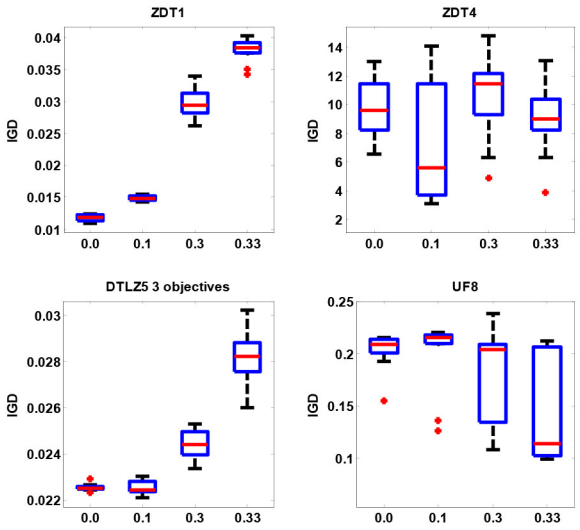


Fig. 4. Effects of *lbound* ($x$-axis) on optimization performance.

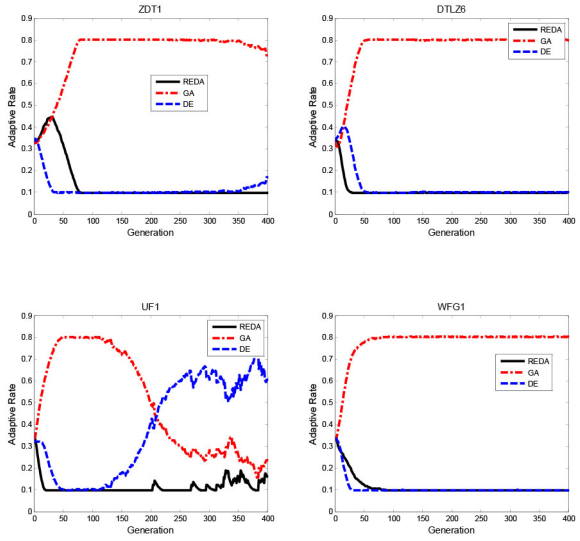


Fig. 3. Adaptive activation of different EAs.

when a larger number of solutions undergoes local search. In UF1, no clear pattern can be observed. In conclusion, the optimal number of solutions to be involved in the local search is problem dependent. However, a smaller number of solutions to undergo local search is better since it may cause extra fitness evaluations in one generation when a large number of solutions is exposed to local search.

### C. Effects of Adaptive Feature on Optimization Performance

Three EAs are considered in the adaptive memetic computing. The adaptive activation of the different EAs is shown in Fig. 3. All plots in the figure show that GA dominates the search at the later stages of evolutions except UF1. The figure also shows that EDA has a higher activation at the early stages of evolutions in ZDT1 and DE has a higher activation at the later stages of evolutions in UF1. These observations suggest that GA is the main algorithm in guiding the search process while DE and EDA aid in exploring some search regions that are not explored by the GA. Since a *lbound* is applied, a population will always consist of solutions generated by different optimizers.

There are two parameters introduced to the adaptive mechanism—*lbound* and learning rate $\epsilon$. *lbound* is important in the case where an algorithm dominates other algorithms at the early stages of evolutions. In this case, the adaptive feature is ceased and the optimization process will only be continued by a single optimizer. Fig. 4 shows the optimization performance in IGD measurement with regards to different settings of *lbound*. Only results for four test problems with different performance are shown. *lbound* $= 0$ means that no *lbound* is set. The figure indicates that the settings of *lbound* gives superior optimization performance in some test problem (e.g., ZDT4), inferior performance in some test problems (e.g., ZDT1), and comparable performance in other test problems (e.g., DTLZ5 with three objectives). *lbound* $= 0.33$ means that the adaptivity is deactivated. The IGD results for *lbound* $= 0.33$ are inferior in most of the test problems (e.g., ZDT1, ZDT4, and DTLZ5 with three objectives) except in UF8.

$\epsilon$ is introduced to the proposed adaptive mechanism to moderate the influences of the proportion of the number of selected solutions in generation $g$ to the whole evolutionary process. Fig. 5 shows the effects of the learning rate on optimization performance. Only results for four test problems with different performances are shown. In ZDT1, a smaller value of $\epsilon$ gives better optimization results. In DTLZ2 with three objectives, a higher value of $\epsilon$ gives better optimization results. In UF8 and WFG2, no clear trend is observed in the settings of $\epsilon$. Overall, a smaller value of $\epsilon$ (0.1–0.5) gives better optimization results in most of the test problems.

### D. Possible Applications

The proposed MAs have advantages on problems with different fitness landscapes. For example, in dynamic optimization problems [54], fitness landscapes are changing over

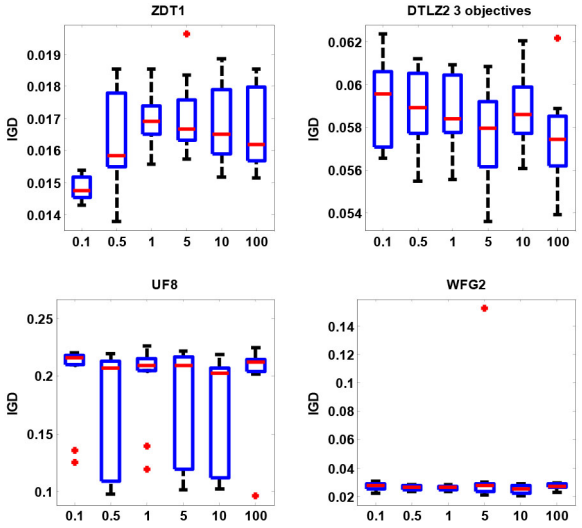


Fig. 5. Effects of learning rate ($x$-axis) on optimization performance.

time either periodically or nonperiodically. Different operators or optimizers may have a better search performance on a particular landscape. The proposed MAs can adaptively use different optimizers to generate offspring upon considering the peculiarity of the fitness landscapes. Some dynamic optimization problems are resource allocation [55] and power scheduling [56].

## VI. Conclusion

The synergetic combination of multiple EAs in an adaptive memetic manner has been proposed in this paper. The memetic feature has leveraged the strengths of each individual EA while the adaptability property has provided a clue to produce a set of promising tradeoff solutions. The proposed technique is a generic synergetic approach to consolidate any EA together such that the mutual genetic information is shared through the propagation of memes into the whole population. The implementation of the adaptive memetic computing in the domination- and decomposition-based MOEAs has not only showed the strengths of the proposed MAs but at the same time provided a deeper comparison between these two frameworks. The validation in a wide range of test problems has proved the efficiency of the proposed algorithms.

## References

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[2] H. A. Abbass and R. Sarker, "The Pareto differential evolution algorithm," *Int. J. Artif. Intell. Tools*, vol. 11, no. 4, pp. 531–552, 2002.

[3] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer, 2001.

[4] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm Evol. Comput.*, vol. 2, pp. 1–14, Feb. 2012.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

[6] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," Caltech Concurrent Computation Program, California Instit. Technol., Pasadena, CA, USA, Tech. Rep. 826, 1989.

[7] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics* (International Series in Operations Research and Management Science). New York, NY, USA: Springer, 2003, pp. 105–144.

[8] X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.

[9] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.

[10] Y. S. Ong, M. Lim, and X. S. Chen, "Memetic computation—Past, present & future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.

[11] F. Neri, G. Iacca, and E. Mininno, "Disturbed exploitation compact differential evolution for limited memory optimization problems," *Inf. Sci.*, vol. 181, no. 12, pp. 2469–2487, 2011.

[12] G. Iacca, F. Neri, E. Mininno, Y. Ong, and M. Lim, "Ockham's razor in memetic computing: Three stage optimal memetic exploration," *Inf. Sci.*, vol. 188, pp. 17–43, Apr. 2012.

[13] F. Caraffini, F. Neri, G. Iacca, and A. Mol, "Parallel memetic structures," *Inf. Sci.*, vol. 227, pp. 60–82, Apr. 2013.

[14] V. Kvasnička and J. Pospíchal, "Simulation of Baldwin effect and Dawkins memes by genetic algorithm," in *Advances in Soft Computing—Engineering Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry, Eds. London, U.K.: Springer-Verlag, 1999, pp. 481–496.

[15] V. A. Shim, K. C. Tan, and K. K. Tan, "A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1142–1149.

[16] J. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," in *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 3, pp. 708–711, 2007.

[17] D. Hadka and P. Reed, "Borg: An auto-adaptive many-objective evolutionary computing framework," *Evol. Comput.*, vol. 21, no. 2, pp. 231–259, 2013.

[18] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[19] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[20] K. Deb and A. Kumar, "Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems," *Complex Syst.*, vol. 9, no. 6, pp. 431–454, 1995.

[21] A. W. Iorio and X. Li, "Solving rotated multiobjective optimization problems using differential evolution," in *Proc. Artif. Intell.*, Cairns, QLD, Australia, 2004, pp. 861–872.

[22] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimization with differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 506–513.

[23] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 561–578, Oct. 2007.

[24] M. Pelikan, K. Sastry, and D. E. Goldberg, "Multiobjective hBOA, clustering, and scalability," in *Proc. 2005 Conf. Genetic Evol. Comput.*, pp. 663–670.

[25] H. J. Tang, V. A. Shim, K. C. Tan, and J. Y. Chia, "Restricted Boltzmann machine based algorithm for multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 3958–3965.

[26] M. Laumanns and J. Ocenasek, "Bayesian optimization algorithms for multi-objective optimization," in *Proc. 7th Int. Conf. Parallel Problem Solving Nature*, Granada, Spain, 2002, pp. 298–307.

[27] Y. Hong, Q. Ren, J. Zeng, and Y. Chang, "Convergence of estimation of distribution algorithms in optimization of additively noisy fitness functions," in *Proc. 17th IEEE Int. Conf. Tools Artif. Intell.*, Hong Kong, 2005, pp. 219–223.

[28] V. A. Shim, K. C. Tan, J. Y. Chia, and A. A. Mamun, "Multiobjective optimization with estimation of distribution algorithm in a noisy environment," *Evol. Comput.*, vol. 20, no. 2, pp. 1–29, 2012.

[29] L. Martí, J. García, A. Berlanga, and J. M. Molina, "Solving complex high-dimensional problems with the multi-objective neural estimation of distribution algorithm," in *Proc. 11th Annu. Conf. Genetic Evol. Comput.*, Montreal, QC, Canada, 2009, pp. 619–626.

[30] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[31] V. A. Shim, K. C. Tan, C. Y. Cheong, and J. Y. Chia, "Enhancing the scalability of multi-objective optimization via restricted boltzmann machine-based estimation of distribution algorithm," *Inf. Sci.*, vol. 248, pp. 191–213, Nov. 2013.
[32] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in *Proc. Soc. Artif. Intell. Statist.*, 2005, pp. 17–22.
[33] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.
[34] S. M. Tse, Y. Liang, K. S. Leung, and K. H. Lee, "A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 84–91, Sep. 2007.
[35] B. Liu, L. Wang, and Y. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 18–27, Feb. 2007.
[36] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 3741–3746.
[37] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength Pareto approach," Comput. Eng. Netw. Lab. (TIK), Swiss Federal Inst. Technol.(ETH) Zurich, Zurich, Switzerland, Tech. Rep. 43, 1998.
[38] R. C. Purshouse and P. J. Fleming, "Evolutionary many-objective optimisation: An exploratory analysis," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 2066–2073.
[39] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 770–784, Dec. 2007.
[40] H. Jain and K. Deb, "An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization," in *Proc. 7th Int. Conf. Evol. Multi-Criterion Optim.*, Sheffield, U.K., 2013, pp. 307–321.
[41] J. Bader and E. Zitzler, "HyPe: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.
[42] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 474–494, Aug. 2013.
[43] H. Wang and X. Yao, "Corner sort for pareto-based many-objective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 92–102, Jan. 2014.
[44] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 203–208.
[45] H. Ishibuchi, T. Yoshida, and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 204–223, Aug. 1998.
[46] K. Deb and T. Goel, "Multi-objective evolutionary algorithms for engineering shape design," *Int. Ser. Oper. Res. Manag. Sci.*, vol. 48, pp. 147–175, 2002.
[47] J. D. Knowles and D. Corne, "Memetic algorithms for multiobjective optimization: Issues, methods and prospects," in *Recent Advances in Memetic Algorithms* (Studies in Fuzziness and Soft Computing), vol. 166. Berlin, Germany: Springer, 2005, pp. 313–352.
[48] J. D. Knowles and D. W. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *Proc. IEEE Congr. Evol. Comput.*, La Jolla, CA, USA, 2000, pp. 325–332.
[49] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 243–259, Apr. 2009.
[50] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[51] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," Kanpur Genetic Algorithms Lab, Indian Inst. Technol., Kanpur, India, KanGAL Rep. 2001001, 2001.
[52] Q. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," University of Essex, Colchester, U.K., Nanyang Technological University, Singapore, Tech. Rep. CES-487, 2009.
[53] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.
[54] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 40–53, Jan. 2014.
[55] J. Branke and D. C. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *Int. J. Product. Res.*, vol. 43, no. 15, pp. 3103–3129, 2005.
[56] K. Deb, N. Udaya Bhaskara Rao, and S. Karthik, "Dynamic multiobjective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," in *Proc. Evol. Multi-Criterion Optim. (EMO)*, Matsushima, Japan, 2007, pp. 803–817.

**Vui Ann Shim** received the B.Eng. degree in electrical and mechatronics engineering from the Universiti Teknologi Malaysia, Malaysia, and the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2008 and 2012, respectively.

He is currently a Scientist with the Institute for Infocomm Research, A*STAR, Singapore. His current research interests include computational intelligence, computational neuroscience, multiobjective optimization, and robotics.

**Kay Chen Tan** received the B.Eng. (Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has published over 100 journal papers, over 100 papers in conference proceedings, co-authored five books, and co-edited four books.

Dr. Tan has served as the General Co-Chair for the IEEE Congress on Evolutionary Computation, Singapore, in 2007, and the General Co-Chair for the IEEE Symposium on Computational Intelligence in Scheduling, Nashville, TN, USA, in 2009. He has been an IEEE Distinguished Lecturer for the IEEE Computational Intelligence Society, since 2011. He is currently the Editor-in-Chief for the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE. He also serves as an Associate Editor and the Editorial Board Member of over 15 international journals. He was a recipient of the 2012 IEEE Computational Intelligence Society Outstanding Early Career Award, the Recognition Award in 2008 from the International Network for Engineering Education and Research, the NUS Outstanding Educator Award in 2004, the Engineering Educator Award in 2002, 2003, and 2005, the Annual Teaching Excellence Award in 2002, 2003, 2004, 2005, and 2006, and the Honor Roll Award in 2007.

**Huajin Tang** received the B.Eng. degree from Zhejiang University, Hangzhou, China, the M.Eng. degree from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 1998, 2001, and 2005, respectively.

He was a Research and Development Engineer with STMicroelectronics, Singapore, from 2004 to 2006. From 2006 to 2008, he was a Post-Doctoral Research Fellow with Queensland Brain Institute, University of Queensland, Brisbane, QLD, Australia. He is currently a Research Scientist with the Institute for Infocomm Research, Singapore. He has published one monograph (Springer-Verlag, 2007) and over 20 international journal papers. His current research interests include neural computation, machine learning, neuromorphic systems, computational and biological intelligence, and neuro-cognitive robotics.

Dr. Tang is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.