A Hybrid Adaptive Evolutionary Algorithm in the Domination-based and Decomposition-based Frameworks of Multi-objective Optimization

V. A. Shim
Department of Electrical and Computer Engineering
National University of Singapore
Singapore
g0800438@nus.edu.sg

K. C. Tan
Department of Electrical and Computer Engineering
National University of Singapore
Singapore
eletankc@nus.edu.sg

K. K. Tan
Department of Electrical and Computer Engineering
National University of Singapore
Singapore
eletankk@nus.edu.sg

Abstract—**Under the framework of evolutionary paradigms, many variations of evolutionary algorithms have been designed. Each of the algorithms performs well in certain cases and none of them are dominating one another. This study is based on the idea of synthesizing different evolutionary algorithms so as to complement the limitations of each algorithm. On top of this idea, this paper proposes an adaptive mechanism that synthesizes a genetic algorithm, differential evolution and estimation of distribution algorithm. The adaptive mechanism takes into account the ratio of the number of promising solutions generated from each optimizer in an early stage of evolutions so as to determine the proportion of the number of solutions to be produced by each optimizer in the next generation. Furthermore, the adaptive algorithm is also hybridized with the evolutionary gradient search to further enhance its search ability. The proposed hybrid adaptive algorithm is developed in the domination-based and decomposition-based multi-objective frameworks. An extensive experimental study is carried out to test the performances of the proposed algorithms in 38 state-of-the-art benchmark test instances.**

Keywords- Decomposition; differential evolution; domination; genetic algorithm; estimation of distribution algorithm; evolutionary gradient search; hybrid multi-objective optimization

I. INTRODUCTION

A multi-objective optimization problem (MOP) is a difficult and complex problem which involves the simultaneous optimization of several conflicting objectives. An MOP can be formulated as follows.

$$\text{Minimize: } F(X) = (f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$$

where X is the decision vector, n is the number of decision variables, F is the objective vector and m is the number of objective functions. $X \in \theta$ and $F \in R^m$ where θ is the decision space and R^m is the objective space. In an MOP, no one solution is optimal to all objectives. Therefore, in order to solve an MOP, search methods employed must be capable of finding a number of alternative solutions representing the trade-off among the various conflicting objectives. A decision vector is optimal if there are no other decision vectors that dominate it in the objective space. The set of all the optimal decision vectors is called the Pareto set (PS) while the set of the corresponding objective vectors forms the Pareto front (PF) [1].

In order to effectively solve an MOP, at least two issues need to be taken into consideration. The first issue is what algorithms are used to explore the search space and the second issue is what frameworks are used to find or maintain the multiple trade-off Pareto optimal solutions. In the algorithmic issue, many multi-objective evolutionary algorithms (MOEAs) have been designed to solve MOPs. For example, MOEAs that use genetic algorithms (GAs) as the search technique are non-dominated sorting genetic algorithm-II (NSGA-II) [2] and MOEA with decomposition (MOEA/D) [3], among others. MOEAs that use differential evolutions (DE) as the search technique are Pareto differential evolution (PDE) [4], generalized differential evolution3 (GDE3) [5] and MOEA/D with DE [6], among others. Next, MOEAs that use estimation of distribution algorithms (EDAs) as the search approach are a multi-objective Parzen-based EDA (MOPED) [7], a regularity model-based multi-objective EDA (RM-MEDA) [8] and a restricted Boltzmann machine based EDA (REDA) [9], among others. Each of the above-mentioned algorithms is efficient in solving certain MOPs and has their own strengths and weaknesses. Furthermore, no evidence indicates that one of the EAs is superior to the others. Thus, it is possible that the synthesis among the EAs can complement their weaknesses while maintaining their strengths.

In the framework's issue, at least three frameworks have been proposed to solve MOPs. They are the aggregation-based, domination-based and decomposition-based frameworks. In the aggregation-based framework, the multiple conflicting objectives of an MOP are aggregated into a single aggregated objective function. Then, a common EA for solving single objective optimization problems is directly applied to solve the aggregated function [10]. However, this framework suffers

two major limitations. First, only one approximate optimal solution can be obtained in a simulation run. Second, it is necessary to specify a weight vector or a preference of a manager for the purpose of aggregation. In order to overcome these shortcomings, the dominance-based framework appears to be more appropriate and efficient.

The domination-based framework is an approach that optimizes all of the objective functions of an MOP simultaneously by assigning a fitness value to each solution where the fitness value is based on the domination behaviours among the solutions in a population [11]. However, a major drawback of this framework is that the selective pressure is weakened with the increase in the number of objective functions. Recently, the decomposition-based framework has been proposed to solve MOPs. This framework decomposes an MOP into several sub-problems. After that, all the sub-problems are optimized concurrently. The selective pressure problem as faced by the domination-based framework does not exist in this framework since the fitness of a solution solely depends on the aggregated objective value. Moreover, it is not necessary to specify a diversity preservation scheme, which is required in the decomposition-based framework, since the diversity can be preserved by using the predefined weight vectors. In the literature, many attempts have been devoted to studying the optimization performances of the algorithms in the domination-based and decomposition-based frameworks. However, the studies that aim to compare the optimization performances of both frameworks are considerably lacking.

This paper has three main aims. First, a GA, DE and EDA are synthesized in an adaptive manner. The adaptive feature takes into account the ratio of the number of promising solutions generated from each optimizer in an early stage of evolutions so as to determine the proportion of the number of solutions to be produced by each optimizer in the next generation. The adaptive algorithm is also hybridized with a local search based on the evolutionary gradient search (EGS). Second, the hybrid adaptive algorithm is developed in the domination-based and decomposition-based frameworks; thus, two hybrid adaptive algorithms are designed. Third, the optimization performances of the proposed algorithms are extensively studied in 38 state-of-the-art benchmark test instances which cover a wide range of characteristics such as deceptive, multi-modal, different shapes of Pareto front, different number of decision variables and objective functions.

The rest of the paper is as follows. Section II provides a thorough literature review to the items that will be covered in this paper. Section III presents the proposed algorithms in both the domination-based and decomposition-based frameworks. Simulation results are presented in section IV. Finally, Section V draws conclusions of this paper.

II. Literature Review

A. Multi-objective Evolutionary Algorithms (MOEAs)

1) MOEAs using GAs

Over the past few decades, many MOEAs that use GAs as the search algorithm have been proposed. NSGA-II [2] is probably one of the most well-known MOEAs. Moreover, MOEA/D [3] is another state-of-the-art MOEAs that has gained significant interest from research communities recently. The genetic operators of both algorithms are based on the simulated binary crossover (SBX) and polynomial mutation. The simulation results obtained in [12] indicated that the SBX operator has the ability to explore the search space when the two selected parents are far from each other, and it is effective in exploiting the promising search regions by generating children solutions that are arbitrary close to the parent solutions. However, it was pointed by Li and Zhang [6] that the SBX operator is not suitable in tackling problems with a complicated Pareto set because the algorithms may loss diversity which is required to explore the search space. In addition, the SBX operator often generates inferior solutions during the evolutionary processes [6]. A detailed description of the SBX operator can be found in [12].

2) MOEAs using DE

DE is another EA that has received significant interest from research communities. Generalized differential evolution (GDE) [13] and its successors, including GDE2 [14] and GDE3 [5], are several famous MOEAs using DE. In GDE, a selection based on Pareto dominance was introduced to a common DE (DE/rand/1/bin) in order to solve MOPs. In GDE2, a crowding distance measurement was incorporated to the original GDE. In GDE3, a non-dominated sorting mechanism, which is used in NSGA-II, and a growing population size were adapted to improve the ability of the GDE in maintaining a set of solutions with good distribution. Another attempt to implement DE to solve MOPs was carried out by Iorio and Li [15]. The authors proposed a non-dominated sorting differential evolution (NSDE), which is directly modified from NSGA-II by replacing the crossover and mutation operators of a GA with the operators of a DE. The simulation results showed that NSDE outperforms NSGA-II in the rotated MOPs. Li and Zhang [6] applied DE into MOEA/D to solve a set of MOPs with a complicated Pareto set. The authors claimed that due to the dissimilarity among the selected parent solutions, DE is able to generate a set of diverse children solutions. However, DE has a difficulty in the functions that are not linearly separable [16]. In addition, DE, in most of the time, will be trapped in local optima in some problem landscapes [17]. A detailed description of the DE's operators can be found in [6].

3) MOEAs using EDAs

Instead of using genetic operators, EDAs employ the probability distribution of the parent solutions to predict the favorable movements of the search processes. More specifically, EDAs construct a probabilistic model of the promising solutions and subsequently generate the children solutions by sampling the constructed model. Any probabilistic modeling approach can be used in EDAs. Some approaches that have been implemented in multi-objective EDAs (MOEDAs) are decision tree [18], local principle

Begin

1. Input: Define initial step size σ_0.

Do while ("Stopping criterion is not met")

For $j = 1$: *LS*(Number of solutions undergoing local search)

2. Initial solution: Select a solution x^j from the selection pool.
3. Reproduction: Create L local neighbors r^i, $i \in (1,2,\dots,L)$ by perturbing x^j using normal mutation $N(0,\sigma^2)$.
4. Evaluation: Calculate the objective values of r^i, $F(r^i)$.
5. Direction: Estimate the global gradient direction as follows.

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

6. Create an offspring:

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

7. Update mutation step size σ_{t+1}:

$$\sigma_{t+1} = \begin{cases} \sigma_t \varepsilon & if\ F(y) < F(x^j) \\ \sigma_t / \varepsilon & otherwise \end{cases}, \varepsilon = 1.8$$

8. Update solution: if $F(y) < F(x^j)$ then $x^j = y$
9. Output: Output x^j.

End for *j*

End do

End

Fig. 1. Pseudo code of an evolutionary gradient search algorithm

component analysis [8], Bayesian network [19], Parzen estimator [7], and restricted Boltzmann machine (RBM) [9]. This paper will only consider an MOEDA that uses RBM (REDA) as its modeling approach.

RBM is an energy-based binary stochastic neural network. The network consists of two layers of neurons. They are a visible layer (denoted as v_i where $i = 1,\dots,n$) and a hidden layer (denoted as h_j where $j = 1,\dots,H$) where n is the number of visible neurons or decision variables and H is the number of hidden neurons. The process flow of REDA consists of three main steps, including the training, modeling and sampling steps. Firstly, the RBM is trained using the contrastive divergence training method [20]. Then, the probability distribution of a visible unit is constructed as follows.

$$Pr(v) = \frac{\sum_h exp\,(-E(v,h))}{\sum_{x,y} exp\,(-E(x,y))}$$

$$Pr(v_i = 1) = \frac{P(v_i^+) + avg\big(P(v_i)\big)}{P\big(v_i^+\big) + P\big(v_i^-\big) + r_i * avg\big(P(v_i)\big)}$$

$$E(v,h) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j$$

where $Pr(v)$ is the probability distribution of a visible unit, $P(v_i^+) = \sum_{h=1}^{m} e^{-E(v_i=1,h)}$ is the marginal cost of v_i when $v_i = 1$, $P(v_i^-) = \sum_{h=1}^{m} e^{-E(v_i=0,h)}$ is the marginal cost of v_i when $v_i = 0$, $avg\big(P(v_i)\big)$ is the average marginal cost of v_i, r_i is the number of cardinalities that v_i may take, E is the energy function of the network, w_{ij} is the weight that connecting visible unit i and hidden unit j, b_i is the bias for the visible unit i and b_j is the bias for the hidden unit j. After constructing the probabilistic model, a child solution is created by sampling the probabilistic model using the simple sampling mechanism [9].

4) Evolutionary gradient search (EGS)

The hybridization of an EA with a local search has been experimentally shown to be able to improve the optimization performance of the search algorithm [21]. The EGS is a local search that exploits the gradient information of the trajectory of solutions and uses it to predict the favorable movements in the search space. This search technique was first developed by Goh and Tan [22] into the multi-objective framework. An archive is created to store all the non-dominated solutions. When the number of solutions in the archive reaches a predefined archive size, a recurrent truncation process is applied to eliminate the most crowded members based on a niche count. This algorithm was extended to study the dynamic MOPs in [23]. The process flow of the EGS is presented in Fig. 1. The ε is set to 1.8 as suggested in [21-23].

B. Algorithmic Frameworks

1) The Domination-based Framework

The domination-based framework aims to optimize all of the objectives of an MOP simultaneously by assigning a fitness value to each solution during the evolutionary processes where the fitness values are based on the domination behaviours among the solutions in a population [24]. Besides, the diversity of the solutions can be maintained through any diversity preservation scheme such as crowding distance or niche sharing. Furthermore, the algorithms in this framework aim to solve MOPs as a whole so that there is no need to specify any preference parameters. NSGA-II, strength Pareto evolutionary algorithm (SPEA) [25] and the Pareto envelope-based selection algorithm (PESA) [26] are among the MOEAs that fall under this category. However, a major drawback of this approach is that the selective pressure is weakened with the increase in the number of objective functions. Furthermore, it is necessary to specify a diversity preservation scheme in order to maintain the diversity of the solution set. Besides, even though a set of diverse solutions exists, the algorithm may not be very good at maintaining the solutions with even distribution along the PF [3].

2) The Decomposition-based Framework

The decomposition-based framework draws its inspiration from the classical multi-objective optimization approaches to decompose an MOP into several scalar optimization sub-problems. The way to transform an MOP into a scalar optimization sub-problem is similar to the aggregation-based framework. In [27], a two-phase local search was proposed. The algorithm aggregates an MOP into a set of scalar optimization sub-problems and then the solutions obtained in the previous sub-problems are used as the starting search point for the next sub-problems.

In [3], Zhang and Li proposed another novel multi-objective optimization algorithm based on decomposition (MOEA/D). This algorithm decomposes an MOP into N scalar optimization sub-problems and subsequently updates the sub-

%%Given a set of selected solutions that are stored in an archive (ψ)

1. Calculate the number of solutions in ψ that are generated from each EA, denoted as $D_g^{EA_i}$ where $i = 1, \dots, M$, M is the number of EAs that are involved in the adaptive process. In this paper, three EAs are involved. Thus, the number of solutions in ψ that are generated from each EA are denoted as $D_g^{EA\_1} = D_g^{GA}$, $D_g^{EA\_2} = D_g^{DE}$ and $D_g^{EA\_3} = D_g^{EDA}$.
2. Calculate the adaptive proportion rate for each EA as follows.
   For $\boldsymbol{i = 1{:}M}$
   $Ar_g^{EA\_i} = Ar_{g-1}^{EA\_i} + \epsilon \times Pr_g^{EA\_i}$, where $Pr_g^{EA\_i} = D_g^{EA\_i}/N$
   End For
   where $Ar_g^{EA\_i}$ is the adaptive proportion rate at generation g for i^{th} EA, ϵ 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.
3. Check for the lower bound of the adaptive proportion rate
   For $\boldsymbol{i = 1{:}M}$
   If $Ar_g^{EA\_i} < l\_bound$
   $Ar_g^{EA\_i} = l\_bound$
   End For
4. Normalize the adaptive proportion rate so that the sum of the adaptive proportion rates is equal to 1.0
   For $\boldsymbol{i = 1{:}M}$
   $Ar_g^{EA\_i} = Ar_g^{EA\_i} / (\sum_{i=1}^{M} Ar_g^{EA\_i})$
   End For

Fig. 2. Pseudo code of the adaptive mechanism

problems by only using the information from its neighboring solutions. MOEA/D optimizes all the sub-problems concurrently instead of directly optimizes an MOP as a whole. MOEA/D has a lower computational complexity compared to NSGA-II and MOGLS [28]. The simulation results in [3] indicated that MOEA/D outperforms MOGLS in 0/1 multi-objective knapsack problems when both of the algorithms use the same decomposition approach. Furthermore, they also found that MOEA/D with an advanced decomposition technique can achieve better convergence and distribution of solutions compared to NSGAII in the continuous MOPs.

III. PROPOSED ALGORITHMS

The fundamental idea of the proposed algorithms in this paper is based on the assumption that combining the different EAs may complement the limitations of each optimizer while maintaining their strengths. On top of this idea, an adaptive feature, which determines the proportion of the number of solutions to be produced by each EA in a generation, is proposed.

Initially, each EA is given an equal chance to produce the initial solutions. After the reproduction processes, a number of promising solutions are selected and stored in an archive. Then, the proportion of the number of solutions to be generated by each optimizer in the next generation is calculated according to the proposed adaptive mechanism as illustrated in Fig. 2. Let ψ be the solutions in an archive. First, calculate the number of solutions in the ψ generated by each EA. Afterward, the adaptive proportion rate ($Ar_g^{EA\_i}$) at generation g for each EA is calculated according to Step 2. A learning rate ($\epsilon < 0$) is incorporated to the updating rule in Step 2 in order to moderate the influences of the proportion of the number of selected solutions in generation g to the whole evolutionary processes. This is because the optimizers that are able to generate a more number of promising solutions in the current generation may not be the superior optimizers in the next generation. In Step 3, a lower bound is set to the adaptive proportion rate. This is necessary since an optimizer may dominate other EAs and finally the adaptive proportion rate of this optimizer will become 1.0 while the adaptive proportion rate of other EAs will become 0.0. When this happens, all children solutions will only be generated by this optimizer till the end of the evolutionary processes. Thus, it is necessary to set a lower bound to the adaptive proportion rate to guarantee that the problem would not exist. Since the summation of all the adaptive proportion rates should be equal to 1.0, the final adaptive proportion rates should be normalized especially when Step 3 is applied (Step 4). Afterward, a typical evolutionary process is continued.

The overall proposed hybrid evolutionary algorithm with non-dominated sorting approach (hNSEA) is presented in Fig. 3. The algorithm starts with a random initialization of an initial population. All of the solutions in the population are evaluated to obtain their corresponding objective values. Next, all solutions are ranked according to the level of domination. For the solutions in the same rank, crowding distance is calculated. Then, select N promising solutions by using the binary tournament selection operator. Subsequently, the adaptive proportion rate for each EA is calculated (Fig. 2). In the reproduction stage, if GA is activated, the SBX and polynomial mutation are used to create an offspring. If DE is activated, the DE operator and polynomial mutation are used to generate an offspring. Similarly, if EDA is activated, then an offspring is sampled from the constructed probabilistic model. After producing N children solutions, evaluation is performed to calculate their objective values. All of the parent and children solutions are stored in an archive. Elitism is performed to select N solutions with the lowest Pareto rank or highest crowding distance from the archive to form the new population. Next, perform the EGS if it is activated. All of the solutions generated from the EGS will also undergo archiving and elitism before forming the new population. A generation is terminated here. The evolutionary processes are continued until the maximum number of fitness evaluations is reached.

The second proposed algorithm is the hybrid MOEA/D (hMOEA/D). hMOEA/D adapts the decomposition algorithm suggested by Li and Zhang [6] and its process flow is illustrated in Fig. 4. This study also uses Tchebycheff approach, as suggested by Li and Zhang [6], to transform an MOP into N scalar optimization sub-problems. Initially, a set of uniformly distributed weight vectors ($\lambda^i, \dots, \lambda^N$) are generated. The Euclidean distance among the weight vectors are calculated. Based on the shortest Euclidean distance, each weight vector is assigned Q neighboring solutions (denoted as $B(i) = \{i_1, \dots, i_Q\}, i \in [1, N]$). Next, an initial population is randomly generated. A reference point for the Tchebycheff approach (z^*) is then initialized to be the lowest objective values of the solutions in the population. After that, the iterative process of the evolution starts. In the reproduction

Begin

1. **Initialization**: At generation $g = 0$, randomly generate N solutions to be the initial population, $Pop(g = 0)$.
2. **Evaluation**: Evaluate each solution in the population.

Do While ("maximum number of fitness evaluations is not reached")

3. **Fitness assignment**: Apply the Pareto ranking and crowding distance over the population. Each solution consists of two values which represent its fitness, one is the rank of the domination and another is the level of crowded.
4. **Selection**: Select N solutions using the binary tournament selection operator.
5. **Adaptive**: Calculate the adaptive proportion rate for each EA (Fig. 2).
6. **Reproduction**: Build a probabilistic model $Pr_g(x)$ to represent the distribution of the solutions (required by REDA).
   For $i = 1:N$
   Generate a random value between 0 and 1 (u)
   If $u < Ar_g^{GA}$
   Generate an offspring using the SBX and polynomial mutation operators.
   Else if $u \geq Ar_g^{GA}$ & $u < Ar_g^{GA} + Ar_g^{DE}$
   Generate an offspring using the DE and polynomial mutation operators.
   Else
   Sample an offspring from $Pr_g(x)$.
   End for
7. **Evaluation**: Calculate the objective values of all children solutions.
8. **Archiving**: Store the parents and children 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 to form the new population.
10. **Local Search**: Generate a random value between 0 and 1 (u).
    If $u < Local\_Search\_Rate$
    For $i = 1:N$
    Generate another random value between 0 and 1 (u)
    If $u < K$ (Percentage of solutions undergoing local search)
    Perform EGS to generate an offspring
    End For
    Perform archiving and elitism to the parents and children solution to form the new population

End Do

End

Fig. 3. Pseudo code of the hybrid non-dominated sorting evolutionary algorithms (hNSEA)

Begin

1. Initialization

a) Generate a set of uniformly distributed weight vectors $(\lambda^i, \dots, \lambda^N)$

b) Calculate the Euclidean distance among the weight vectors. Determine the Q neighboring solutions $(B(i) = \{i_1, \dots, i_Q\}, i \in [1, N])$ for each weight vectors according to the shortest Euclidean distance.

c) At generation g=0, randomly generate N solutions to be the initial population, $Pop(g = 0)$

d) Initialize reference point of the Tchebycheff approach (z^*) by setting the value of z^* to be the lowest objective values of the solutions

Do while ("maximum number of fitness evaluations is not reached")

Build a probabilistic model of the whole population $Pr_g(x)$ to represent the distribution of the solutions (required by REDA)

For $i = 1:N$

2. **Reproduction**: Generate a random value between 0 and 1 (u)
   If $u < Ar_g^{GA}$
   Randomly select two solutions from $B(i)$, and then generate an offspring using the SBX and polynomial mutation operators
   Else if $u \geq Ar_g^{GA}$ & $u < Ar_g^{GA} + Ar_g^{DE}$
   Randomly select three solutions from $B(i)$, and then generate an offspring using the DE and polynomial mutation operators
   Else
   Build another probabilistic model $Pr_g'(x)$ which only considers the neighboring solutions. Sampling an offspring: Generate a random value between 0 and 1(u)
   For $j = 1:n$ (number of decision variables)
   if $(u < 0.5)$ Sample from $Pr_g(x)$
   else Sample from $Pr_g'(x)$
   End For
3. **Evaluation**: Evaluate the generated offspring (y) to obtain the corresponding objective values, $f(y)$
4. **Update of z^*:** For $j = 1, \dots, m$, if $z_j^* > f_j(y)$, then set $z_j^* = f_j(y)$
5. **Fitness assignment**: Assign fitness to each solution (g^{te}) using Tchebycheff method
6. **Update Solution**: For $j \in B(i)$, if $g^{te}(y|\lambda^j, z^*) \leq g^{te}(x^j|\lambda^j, z^*)$, then set $x^j = y$ and $FV^j = F(y)$

End For

7. **Local search:** Perform the EGS if it is activated (similar to Fig. 3. Step 10). Then, apply Steps 4-6 to update the reference point and the neighboring solutions

End Do

End

Fig. 4. Pseudo code of the hybrid MOEA/D (hMOEA/D)

stage, if GA is activated, then randomly select two neighboring solutions of solution i to undergo the SBX and polynomial mutation. If DE is activated, p_1 is set to be the solution i and then randomly select another two neighboring solutions of solution i. A child solution is generated by the DE operator and polynomial mutation with a probability of p_m. If EDA is activated, then generate a random value between 0 and 1 in order to determine which probabilistic models are used to sample an allele of the child solutions. After a child solution was created, the solution is evaluated to obtain its objective values. Next, the reference point (z^*) is updated. A fitness value is assigned to this solution using the Tchebycheff approach. Then, update the neighboring solutions if the child solution is fitter than the neighboring solutions. Finally, perform the EGS if it is activated. A generation is completed here. The iterative process continues until the maximum number of fitness evaluations is reached.

IV. Experimental Results

A. Implementations

Eight algorithms were involved in the experimental studies. All of the algorithms were implemented in C++. The algorithms are NSGA-II with SBX (NSGAII-SBX) [2], NSDE [15], MOEA/D with SBX (MOEAD-SBX) [3], MOEA/D with DE (MOEA/D-DE) [6], NSREDA [9], MOEA/D with REDA (MOEA/D-REDA), hNSEA and hMOEA/D. The first four algorithms are the state-of-the-art algorithms of multi-objective optimization. NSREDA is a recently developed EDA that uses RBM as its modeling approach. In this paper, we adapt the NSREDA into the decomposition-based framework (MOEA/D-REDA). hNSEA and hMOEA/D are the proposed hybrid adaptive algorithms. The parameter settings of the algorithms are presented in Table I. Thirty-one test instances with two or three objective functions (ZDT problems [29], DTLZ problems [30], UF problems [31], and WFG problems [32]) plus seven test instances with five objective functions (DTLZ problems) are used to test the optimization performances of the proposed algorithms. The performance metric of the inverted generational distance (IGD) [8] is used to evaluate the optimization performances of the algorithms. IGD measures the Euclidean distance of the solutions from the Pareto optimal front to the evolved front. A smaller value of IGD indicates a better optimization performance.

B. Comparison Results

Table II shows the optimization results in terms of IGD measurement generated from the various algorithms. The first

TABLE I. PARAMETER SETTINGS

Population size	100 for problems with two objective functions, 300 for problems with three objective functions, and 500 for problems with five objective functions.
Stopping criterion (number of fitness evaluations)	500 × Population size
Number of runs	10
Lowe bound in the adaptive feature	0.1
Learning rate in the adaptive feature	0.1
Number of hidden units in REDA (H)	5
Number of training epochs in REDA	2
Distribution index in the SBX and polynomial mutation (φ)	20
Mutation rate (p_m)	0.8
DE and crossover rate of the SBX	0.9
Local search rate ($Local\_Search\_Rate$)	0.5
Number of solutions to undergo local search (K)	10% from the population size
Number of local neighbor (L)	4
Number of neighboring solutions in the decomposition-based algorithms (Q)	20

column in the table indicates the test problems. The parentheses next to the test problems refer to the number of decision variables (n) and objective functions (m) of the test problems. The average IGD values over 10 simulation runs are tabulated. The numbers inside the parentheses next to the IGD values refer to the ranking of the algorithms in a specific test instance. All the rankings performed in this section are based on the scores inside the parentheses.

ZDT problems are a set of simple MOPs that possess two objective functions and a scalable number of decision variables. Since all of the algorithms can easily solve the ZDT problems, we set the number of decision variables 10 times greater than its original setting. Thus, the ZDT problems have a large search space. The simulation results indicate that hNSEA has the best performance, followed by hMOEA/D. However, both of the algorithms fail to converge to the PF in ZDT4. This is because ZDT4 is a multimodal problem which consists of many local optima. The results also show that EDA is able to generate a set of good results in ZDT problems, followed by GA and DE. In terms of the framework's issue, the domination-based algorithms generate better results than the decomposition-based algorithms. The ranking of the algorithms in ZDT problems is hNSEA, hMOEA/D, NSREDA, MOEA/D-REDA, NSGA-II-SBX, MOEA/D-SBX, MOEA/D-DE and NSDE.

DTLZ problems consist of a scalable number of objective functions and decision variables. DTLZ1 and DTLZ3 are multimodal test problems. Due to the difficulties of both problems, we set the number of decision variables to be 12. For the other DTLZ problems that are easier to solve, the number of decision variables is set to 120. The results indicate that, in DTLZ problems with three objective functions, hNSEA has the best results in DTLZ5 and DTLZ6 where the problems have degenerate PF. hMOEA/D has the best performances in DTLZ2 and DTLZ3 where the problems have spherical shape of PF. For problems with multi-modality (DTLZ2 and DTLZ3), the hybrid algorithms and the algorithms with DE show better performances compared to the algorithms with GA and EDA. This finding is identical to the results obtained in [9] which concluded that EDAs are easily trapped in local optima. In DTLZ7 (problem with discontinuity PF), NSREDA shows the best performance. In terms of the framework's issue, the decomposition-based algorithms have better IGD results than the domination-based algorithms except hybrid algorithms where hNSEA has smaller IGD values than hMOEA/D. The ranking of the algorithms in DTLZ problems with three objective functions is hNSEA, hMOEA/D, MOEA/D-SBX, NSGA-II-SBX, MOEA/D-DE, MOEA/D-REDA, NSREDA and NSDE.

In DTLZ problems with five objective functions, the simulation results indicate that hMOEA/D and MOEAD-SBX have the best results. Even though hNSEA is able to obtain the best results in DTLZ3 and DTLZ6, its performances in other DTLZ problems are poor. The decomposition-based algorithms show better IGD results than the domination-based algorithms. The ability of the decomposition-based algorithms in solving problems with many objective functions has been discussed in Section II that the decomposition-based framework can differentiate the superiority of the solutions by using the aggregated fitness values while the decomposition-based algorithms need to determine the domination behaviours among the solutions before the superiority of the solutions is determined, where the domination behaviours is weakened with the increase in the number of objective functions. The experimental results also show that the performances of EDA are superior to GA and DE. This finding is consistent with the results reported in [9]. The ranking of the algorithms in DTLZ problems with five objective functions is hMOEA/D, MOEA/D-SBX, MOEA/D-REDA, hNSEA, NSREDA, NSGA-II-SBX, MOEA/D-DE and NSDE.

UF problems can be regarded as a set of MOPs with complicated PS shapes. It is a set of difficult MOPs that were proposed for the CEC 2009 competition. We preserve the number of decision variables to be 30 even though they are scalable. UF1-UF7 consist of two objective functions and UF8-UF10 possess three objective functions. The simulation results show that hNSEA obtains the best IGD values in six UF test problems, with hMOEA/D obtaining two best results and MOEA/D-SBX and MOEA/D-DE obtaining one best results respectively. The MOEAs with EDA give inferior solutions to most of the UF test problems. This may be cause by the fact that the EDA fails to construct a probabilistic model which is able to represent the complicated distribution of the solutions in the decision space. Besides, the ability of the algorithms in generating a set of diverse solutions is critical in addressing the UF problems [6]. In [18, 33], the researchers claimed that EDA is particularly weak in generating a set of diverse solutions since only global information of the probability distribution is used. In terms of the framework's issue, there is no clear superiority in the performances for both the domination-based and decomposition-based algorithms. The ranking of the algorithms in UF problems is hNSEA, hMOEA/D, MOEA/D-

TABLE II. IGD VALUES GENERATED BY THE VARIOUS ALGORITHMS.

Test instance (m,n)	NSGAII-SBX	MOEAD-SBX	NSDE	MOEAD-DE	NSREDA	MOEAD-REDA	hNSEA	hMOEAD
ZDT1(2,300)	0.1979 (5)	0.2997 (6)	1.3690 (8)	1.0675 (7)	0.1497 (3)	0.1895 (4)	**0.0148 (1)**	0.0274 (2)
ZDT2(2,300)	0.3791 (5)	0.4747 (6)	2.7422 (8)	1.9065 (7)	0.2715 (3)	0.3521 (4)	**0.0161 (1)**	0.0323 (2)
ZDT3(2,300)	0.1585 (3)	0.3141 (6)	0.9509 (8)	0.9205 (7)	0.1696 (4)	0.2016 (5)	**0.0126 (1)**	0.0781 (2)
ZDT4(2,100)	25.3201 (4)	29.0530 (6)	28.4168 (5)	35.3880 (7)	18.2170 (3)	16.2532 (2)	**7.3029 (1)**	36.7845 (8)
ZDT6(2,100)	0.9203 (4)	0.6175 (3)	5.9244 (8)	3.7106 (7)	3.1699 (6)	3.0413 (5)	**0.0103 (1)**	0.4245 (2)
DTLZ1(3,12)	0.6809 (6)	0.0215 (5)	**0.0137 (1)**	0.0173 (4)	40.5211 (8)	8.4998 (7)	0.0141 (2)	0.0172 (3)
DTLZ2(3,120)	0.0458 (3)	0.0327 (2)	2.0191 (8)	0.0659 (5)	1.5441 (7)	0.0776 (6)	0.0571 (4)	**0.0319 (1)**
DTLZ3(3,12)	5.3465 (6)	0.0353 (3)	0.0372 (4)	**0.0330 (1)**	131.4790 (8)	28.8747 (7)	0.0392 (5)	**0.0330 (1)**
DTLZ4(3,120)	**0.0529 (1)**	0.2183 (3)	12.4117 (8)	12.0419 (7)	1.6450 (6)	1.0396 (5)	0.2779 (4)	0.1535 (2)
DTLZ5(3,120)	0.0229 (2)	0.0240 (3)	1.0776 (7)	0.0274 (5)	1.6679 (8)	0.0377 (6)	**0.0225 (1)**	0.0240(3)
DTLZ6(3,120)	48.0578 (7)	21.1870 (5)	57.137 (8)	3.1601 (4)	2.9239 (3)	0.0246 (2)	**0.0235 (1)**	26.0278 (6)
DTLZ7(3,120)	0.1008 (3)	0.1742 (5)	2.7655 (8)	0.4761 (7)	**0.0867 (1)**	0.1679 (4)	0.0939 (2)	0.1743 (6)
UF1(2,30)	0.0583 (3)	0.0639 (4)	0.1174 (5)	0.1248 (7)	0.1268 (8)	0.1181 (6)	0.0549 (2)	**0.0441 (1)**
UF2(2,30)	0.0443 (3)	0.0520 (6)	0.0447 (4)	0.0482 (5)	0.1063 (8)	0.0654 (7)	0.0383 (2)	**0.036 (1)**
UF3(2,30)	0.1307 (2)	0.1437 (3)	0.2517 (5)	0.3069 (6)	0.3706 (8)	0.3088 (7)	**0.1292 (1)**	0.1867 (4)
UF4(2,30)	0.0731 (5)	0.0807 (7)	0.0526 (3)	0.0536 (4)	0.1355 (8)	0.0910 (6)	**0.0507 (1)**	0.0520 (2)
UF5(2,30)	1.0556 (8)	0.7361 (7)	0.3232 (2)	0.4508 (4)	0.4727 (5)	0.4930 (6)	**0.3163 (1)**	0.4157 (3)
UF6(2,30)	0.0552 (3)	0.0547 (2)	0.1602 (5)	0.1617 (6)	0.1734 (8)	0.1629 (7)	**0.0437 (1)**	0.1129 (4)
UF7(2,30)	0.0273 (2)	0.1132 (4)	0.2229 (5)	0.3157 (7)	0.3935 (8)	0.2988 (6)	**0.0249 (1)**	0.1075 (3)
UF8(3,30)	0.1400 (4)	**0.0843 (1)**	0.2193 (8)	0.1310 (3)	0.2109 (7)	0.1937 (5)	0.1994 (6)	0.1141 (2)
UF9(3,30)	0.1611 (5)	0.1182 (2)	0.1671 (6)	**0.0862 (1)**	0.4152 (8)	0.3661 (7)	0.1287 (3)	0.1378 (4)
UF10(3,30)	2.5149 (8)	0.6143 (7)	0.3062 (2)	0.3163 (3)	0.4578 (5)	0.488 (6)	**0.2906 (1)**	0.4001 (4)
WFG1(2,30)	1.2696 (7)	1.2312 (6)	1.3732 (8)	1.2165 (5)	1.1610 (3)	1.1832 (4)	0.9771 (2)	**0.9362 (1)**
WFG2(2,30)	0.0308 (2)	0.0650 (3)	0.1019 (6)	0.1429 (8)	0.0879 (4)	0.1377 (7)	**0.0275 (1)**	0.097 (5)
WFG3(2,30)	**0.0244 (1)**	0.0282 (5)	0.0254 (2)	0.0274 (4)	0.0630 (7)	0.0959 (8)	0.0273 (3)	0.0320 (6)
WFG4(2,30)	0.1004 (8)	0.0845 (7)	0.0189 (3)	**0.0150 (1)**	0.0284 (5)	0.0506 (6)	0.0204 (4)	0.0155 (2)
WFG5(2,30)	0.0734 (6)	0.0673 (5)	0.0669 (3)	**0.0663 (1)**	0.0759 (8)	0.0756 (7)	0.0699 (3)	0.0668 (2)
WFG6(2,30)	0.0869 (8)	0.0875 (7)	0.0467 (3)	0.0466 (2)	0.0602 (5)	0.0693 (6)	0.0484 (4)	**0.0362 (1)**
WGG7(2,30)	0.0333 (6)	0.0181 (4)	0.0168 (3)	0.0142 (2)	0.0364 (7)	0.0457 (8)	0.0182 (5)	**0.0141 (1)**
WFG8(2,30)	0.1215 (6)	0.1109 (5)	0.0797 (3)	0.0763 (2)	0.1431 (8)	0.1261 (7)	0.0826 (4)	**0.0741 (1)**
WFG9(2,30)	0.0332 (6)	0.0291 (5)	0.0210 (3)	0.0179 (2)	0.0460 (8)	0.0384 (7)	0.0215 (4)	**0.0165 (1)**
DTLZ1(5,12)	22.5028 (6)	0.2176 (2)	62.3703 (7)	260.9400 (8)	4.9827 (4)	3.7365 (3)	12.3594 (5)	**0.2145 (1)**
DTLZ2(5,120)	1.5922 (5)	1.0260 (2)	3.9636 (6)	1.0535 (4)	10.4467 (8)	**0.9940 (1)**	8.3581 (7)	1.0302 (3)
DTLZ3(5,12)	75.4146 (6)	1.0020 (3)	217.494 (7)	960.3300 (8)	12.1573 (4)	17.9906 (5)	**0.9197 (1)**	0.9335 (2)
DTLZ4(5,120)	6.0306 (4)	**1.1211 (1)**	21.1929 (6)	9.2750 (5)	25.1628 (8)	4.5550 (3)	24.8215 (7)	1.7985 (2)
DTLZ5(5,120)	4.5888 (6)	0.9331 (2)	1.3406 (5)	0.9375 (3)	9.1865 (8)	0.9369 (4)	7.7750 (7)	**0.9307 (1)**
DTLZ6(5,120)	96.6994 (8)	15.5819 (5)	80.6766 (7)	10.3338 (4)	20.2494 (6)	0.9297 (2)	**0.9008 (1)**	0.9307 (3)
DTLZ7(5,120)	4.8243 (5)	4.2058 (3)	9.0199 (7)	11.9862 (8)	**3.6190 (1)**	4.3186 (4)	4.1183 (2)	6.1212 (6)

SBX, NSGA-II-SBX, NSDE, MOEA/D-DE, MOEA/D-REDA and NSREDA.

WFG problems are another set of difficult MOPs that involve various types of transformations. The problems consist of a scalable number of objective functions and decision variables. In this paper, two objective functions and 30 decision variables are applied. The decision vector consists of two position parameters and 28 distance parameters. The simulation results show that hMOEA/D generates a set of solutions with the best IGD values in five WFG problems. The performances of the algorithms with DE are better than the algorithms with GA and EDA. In terms of the framework's issue, the decomposition-based algorithms outperform the domination-based algorithms slightly. The ranking of the algorithms in WFG problems is hMOEA/D, MOEA/D-DE, hNSEA, NSDE, MOEA/D-SBX, NSGA-II-SBX, MOEA/D-DE, NSREDA and MOEA/D-REDA.

Overall, hNSEA obtains the best IGD results in 16 test problems followed by hMOEAD with the best IGD results in 12 test problems. These findings demonstrate that the proposed hybrid adaptive mechanism improves the optimization performance of an individual optimizer. However, it also happens that the hybrid adaptive algorithms generate the worse IGD values, as indicate in ZDT4, DTLZ2 and DTLZ4 with five objective functions. To sum up, the hybrid adaptive mechanism proposed in this paper succeeds in complementing the limitations of an individual EA and in maintaining their search abilities in most of the test instances. In terms of individual EA, the performances of GA are superior in DTLZ problems with three objective functions and UF problems. Its performances in ZDT, DTLZ with five objective functions and WFG problems are average. The performances of DE are superior in WFG problems, average in DTLZ problems with three objective functions and UF problems and inferior in ZDT and DTLZ problems with five objective functions. The performances of EDA are superior in ZDT and DTLZ problems with five objective functions and inferior in the other MOPs. In terms of the framework's issue, the domination-based algorithms are superior to the decomposition-based algorithms in ZDT problems, comparable in UF test problems and inferior in the other test problems. The overall ranking of the algorithms in all the test problems is hNSEA, hMOEA/D, MOEA/D-SBX, MOEA/D-DE, NSGA-II-SBX, MOEA/D-REDA, NSDE and NSREDA.

V. CONCLUSIONS

This study proposed an adaptive mechanism to synthesize a GA, DE and EDA for multi-objective optimization. The

adaptive mechanism takes into account the ratio of the number of promising solutions generated by each optimizer in an early stage of evolutions so as to determine the proportion of the number of solutions to be produced by each optimizer in the next generation. The search ability of the adaptive algorithm was further enhanced by hybridizing it with a local search based on the EGS. The hybrid adaptive algorithm was constructed in the domination-based and decomposition-based frameworks of multi-objective optimization. The performances of the hybrid adaptive algorithms, together with other six state-of-the-art MOEAs, were tested in 38 benchmark test instances. The simulation results showed that the proposed algorithms outperform the other six algorithms. The results also demonstrated that the optimization performances of the decomposition-based algorithms are clearly superior to the decomposition-based algorithms in the test problems with five objective functions.

REFERENCES

[1] K. Deb, Multi-objective Optimization using Evolutionary Algorithms, Chichester John Wiley & Sons, 2001.

[2] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[3] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, 2007.

[4] H. A. Abbass and R. Sarker, "The Pareto differential evolution algorithm," *International Journal of Artificial Intelligence Tools*, vol. 11, no. 4, pp. 531-552, 2002.

[5] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *proceeding of IEEE Congress on Evolutionary Computation* , pp. 443-450, 2005.

[6] H. Li and Q. Zhang, "Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284-302, 2009.

[7] M. Costa and E. Minisci, "MOPED: a multi-objective Parzen-based estimation of distribution algorithm for continuous problems," in *Proceedings of Evolutionary Multi-Criterion Optimization*, pp. 282-94, 2003.

[8] Q. Zhang, A. Zhou, and Y. Ji, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41-63, 2008.

[9] H. J. Tang, V. A. Shim, K. C. Tan, and J. Y. Chia., "Restricted Boltzmann machine based algorithm for multi-objective optimization," *IEEE Congress on Evolutionary Computation*, pp. 1-8, 2010.

[10] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization, Ph.D, Swiss Federal Institute of Technology, Zurich, 1999.

[11] D. E. Goldberd, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[12] K. Deb and A. Kumar, "l-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems," *Complex System*, vol. 9, pp. 431-454, 1995.

[13] J. Lampinen, "DE's selection rule for mul-tiobjective optimization," Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001.

[14] S. Kukkonen and J. Lampinen "An extension of Generalized Differential Evolution for multi-objective optimization with constraints," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, pp. 752-761, 2004.

[15] A. W. Iorio and X. Li, "Solving rotated multiobjective optimization problems using differential evolution," in *Procdeeding of Artificial Intelligence: Advances in Artificial Intelligence*, pp. 861-872, 2004.

[16] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimization with differential evolution," in *Proceeding of IEEE congress on evolutionary computation*, pp. 506-513, 2005.

[17] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms*," IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 561-578, 2007.

[18] X. Zhong, and W. Li, "A decision-tree-based multi-objective estimation of distribution algorithm," *Proceedings of the International Conference on Computational Intelligence and Security*, pp. 114-118, 2007.

[19] M. Pelikan, K. Sastry, and D. E. Goldberg, "Multiobjective hBOA, clustering, and scalability," *Genetic and Evolutionary Computation Conference*, pp. 663-670, 2005.

[20] G. E. Hinton, "What kind of a graphical model is the brain?," *Proceedings of the international joint conference on Artificial intelligence*, pp. 1765-1775, 2005.

[21] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp.42-50, 2007.

[22] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization", *IEEE Congress on Evolutionary Computation*, pp. 3741-3746, 2008.

[23] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment", *Memetic Computing*, vol. 2, no. 2, pp. 87-110, 2010.

[24] D. E. Goldberd, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[25] Zitzler, E. and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland, 1998.

[26] D Corne, J Knowles, M Oates, "The Pareto-Envelope based Selection Algorithm for multiobjective optimization," *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature*, pp. 839-848, 2000.

[27] L. Paquete and T. Stutzle, "A two-phase local search for the biobjective traveling salesman problem," in *Proceeding of Evolutionary Multi-Criterion Optimization*, pp. 479-493, 2003.

[28] A. Jaszkiewicz, "On the performance of multi-objective genetic local searhon the 0/1 knapsack problem - A comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117-132, 2003.

[29] E. Zitzler, ''Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications,'' Swiss Federal Institute of Technology, Zurich, 1999.

[30] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, ''Scalable multiobjective optimization test problems,'' in *Proceedings of the Congress on Evolutionary Computation. CEC,* pp. 825-30, 2002.

[31] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, ''Multiobjective optimization test instances for the CEC 2009 special session and competition,'' Technical Report CES-487, 2009.

[32] S. Huband, P. Hingston, L. Barone, and L. While, ''A review of multiobjective test problems and a scalable test problem toolkit,'' *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, 2006.

[33] V. A. Shim, K. C. Tan, J. Y. Chia, and J. K. Chong, "Evolutionary algorithms fr solving multi-objective traveling salesman problem," *Flexible Service and Manufacturing Journal*, vol. 23, no. 2, pp. 207-241, 2011.