Dynamic economic emission dispatch through evolutionary multiobjective optimization: An experimental study

Hsuan-Jen Ko
Dept. of Computer Science and Information Engineering
National Taiwan Normal University
Taipei, Taiwan, R. O. C.
willy9966@gmail.com

Thammarsat Visutarrom
Dept. of Computer Science and Information Engineering,
National Taiwan Normal University
Taipei, Taiwan, R. O. C.
thammarsat@gmail.com

Tsung-Che Chiang
Dept. of Computer Science and Information Engineering,
National Taiwan Normal University
Taipei, Taiwan, R. O. C.
tcchiang@ieee.org

<< This paper is included in the Proceedings of IEEE Symposium Series on Computational Intelligence (SSCI), Singapore, 2022. >>

***Abstract*—This paper addresses the dynamic economic emission dispatch (DEED) problem, which needs to allocate power output of generation units in a power system to satisfy power demand over consecutive time periods and minimize cost and emissions simultaneously. We propose a multiobjective evolutionary algorithm to solve the DEED problem. We investigate the effects of the algorithm components including the repair mechanism, selection mechanism, dynamic resource allocation, and dynamic mutation through comprehensive experiments on six test cases. We also compare our algorithm with 15 existing algorithms, and our algorithm shows competitive performance.**

Keywords—economic dispatch, emissions, multiobjective, constraint handling, evolutionary algorithm

I. INTRODUCTION

Energy management and environmental protection are important issues nowadays. Many real-world problems start to take related objectives into consideration. For example, production scheduling problems consider energy consumption and vehicle routing problems consider pollution emissions. Economic dispatch (ED) is a problem about power allocation in a power generating system, and the goal is to minimize the generation cost. Economic emission dispatch (EED) extends the ED problem by considering pollution emissions to meet the concern of environmental protection. In this paper, we deal with the dynamic economic emission dispatch (DEED) problem, which solves EED problems in consecutive time periods.

In a DEED problem, we are given N generation units. We need to decide the power generated by each unit $P_{i,t}$ (i = 1, 2, …, N; t = 1, 2, …, T) so that the total power can satisfy the required power demand in each time period t. Besides, the generation cost and pollution emissions must be minimized. Technically, the problem can be formulated as (1)–(6) [1].

$$f_1: f_{cost} = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[a_i + b_i P_{i,t} + c_i P_{i,t}^2 + \left|d_i\{\sin(e_i(P_i^{min} - P_{i,t}))\}\right|\right] \tag{1}$$

$$f_2: f_{emission} = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[\alpha_i + \beta_i P_{i,t} + \gamma_i P_{i,t}^2 + \eta_i \exp(\delta_i P_{i,t})\right] \tag{2}$$

Equations (1) and (2) present the two objective functions, the cost and emissions. Values of all variables except $P_{i,t}$ are known and static. They could be different in different systems. The DEED problem has three types of constraints. The power limit constraints are presented in (3).

$$P_i^{min} \le P_{i,t} \le P_i^{max} \tag{3}$$

The power balance constraint (4) presents that in each time period t the total generated power must be equal to the total power demand $P_{D,t}$. Here, we follow the Kron's formula [1] to calculate the power loss $L(P_t)$ by (5). When the model does not consider power loss, $L(P_t)$ is set by zero.

$$\sum_{i=1}^{N} P_{i,t} = P_{D,t} + L(P_t) \tag{4}$$

$$L(P_t) = \sum_{i=1}^{N}\sum_{j=1}^{N} P_{i,t} B_{ij} P_{j,t} + \sum_{i=1}^{N} P_{i,t} B_{0i} + B_{00} \tag{5}$$

The DEED problem is more realistic than the EED problem by considering different power demand in consecutive time periods. When the demand changes, the power generation units will adjust their power output. This brings the third type of constraints in DEED – the ramp rate limit constraints, as shown in (6). These constraints present that the power output of each generation unit could not change too much at a time. DR_i and UR_i refer to the maximal decrement and increment of power for generation unit i between two consecutive time periods t–1 and t.

$$P_{i,t-1} - DR_i \leq P_{i,t} \leq P_{i,t-1} + UR_i \tag{6}$$

The DEED problem is a real-parameter, multiobjective, and constrained optimization problem. In this paper we address this problem by a decomposition-based multiobjective evolutionary algorithm. Our algorithm is able to find a set of non-dominated solutions in a single run; in other words, users do not need to try different objective weights and run the algorithm several times to find a satisfactory solution. We add a repair mechanism to cope with constraints and ensure all solutions are feasible. The proposed algorithm adaptively allocates computational resource to sub-problems and mutation operators to search more effectively. Benefits of the algorithm components are verified through comprehensive experiments. The whole algorithm also shows good performance when comparing with 15 existing algorithms.

The rest of this paper is organized as follows. Section II reviews the literature on DEED. Section III details the proposed algorithm. Experimental results and discussions are given in Section IV. Section V concludes this research and lists future work.

II. Literature Review

In this section we review the existing works on the DEED problem, particularly those applying evolutionary algorithms and metaheuristics. Evolutionary algorithms solve problems through an iterative process of generating solutions and selecting good solutions. Thus, in the first two sub-sections we review the literature by focusing how they generated solutions and how they selected solutions respectively. Most operators of evolutionary algorithms need parameters, and parameter control is an important topic. Thus, in the last sub-section we review some parameter control methods in the literature.

A. Solution Generation

Basu [1][2] proposed the earliest studies on the DEED problem. In [1], he solved the DEED problem based on particle swarm optimization (PSO). PSO generates solutions in a way like birds seeking for foods and moving by considering the difference between their own positions and local and global best positions. In [2], Basu used NSGA-II [3], which is a well-known multiobjective evolutionary algorithm. Following the design of NSGA-II, the simulated binary crossover and polynomial mutation operators [4] were adopted to produce solutions. As another good candidate for solving continuous optimization problems, differential evolution (DE) was taken as the main algorithm framework by Jiang *et al*. [5]. They used the classic rand/1/bin strategy. Mason *et al*. [6] did some modifications on the PSO operator. First, they added a move-away velocity from the worst solution; second, they gradually increased the neighborhood size of particles so that exploration is stronger at the beginning of the search process and exploitation is stronger at the later stage. In MONNDE [7], Mason *et al*. used a neural network (NN) to produce solutions. They evolved the topology of networks by genetic algorithm and evolved the weights of neuron connections by DE. The inputs of the NN are the objective weights and the power demand of current and previous time periods, and the outputs are the power generated by the generation units.

Since different reproduction operators may have different features and abilities, it is a natural attempt to combine multiple operators. In [8], Guo *et al*. adopted group search optimizers (GSO), in which solutions were classified into different roles: producers, scroungers, and rangers. Producers and rangers moved based on angles and a polar-to-Cartesian coordinate transformation, while scroungers moved based on the difference between solutions like PSO and DE. Roy and Bhui [9] hybridized chemical reaction optimization (CRO) and DE. Their algorithm was based on CRO, which had four kinds of operators to produce solutions. The crossover operators of DE was carried out before applying the CRO operators. Shen *et al*. [10] applied DE and selected between rand/1 and best/1 mutation operators based on the population similarity. The rand/1 operator was applied in a higher probability when the population similarity is lower. Qian *et al*. [11] combined PSO and DE. To increase the local search ability, they applied clone selection to pick up good solutions and then applied either DE/rand/1 or DE/best/1 mutation operators to these solutions in equal probability. In addition to combining operators of metaheuristics, mathematical optimization methods were also tried in the literature. For example. Elaiw *et al*. [12] used DE or PSO for global search and sequential quadratic programming (SQP) for local search.

B. Selection and Diversity Control

The DEED problem aims to minimize two objective functions, cost and emissions, simultaneously. Since the two objective functions might be in conflict, it is necessary to design a proper method to compare and select solutions during the search process. The simplest method is to aggregate two objective values into one value through weighted summation and then compare solutions based on the aggregated value [10][11]. By using a set of weight values, an approximation set of Pareto optimal solutions could be obtained [7]. NSGA-II has been widely used in solving multiobjective optimization problems, and its selection method was adopted in several studies on DEED [2][13]. Solutions are classified into several fronts by non-dominated sorting and are compared by their level of non-domination. Solutions of the same level are compared by the crowding distance, which is a measure of distance between neighboring solutions in the objective space. Some studies modified the dominance relationship [5] or the distance measure [14]. Zhu *et al*. [15] proposed to calculate the crowding distance dynamically and control the number of solutions of each front that can survive to the next generation. In addition to objective aggregation and dominance-based methods,

the third method is through sub-problem decomposition. MOEA/D [16] is a representative decomposition-based algorithm and was applied by Zhu *et al*. [17] to solve the DEED problem.

Selection helps the search process to converge, but premature convergence may degrade the solution quality. A simple way to increase population diversity is carrying out more perturbation onto the solutions, for example, using polynomial mutation [17] or levy flight operator [18]. Another method is through re-initialization. One example is that Jiang *et al*. [5] replaced the worse-half solutions of the current generation by new random solutions. The range of random values of power output was controlled and usually became smaller gradually during the search process.

C. Parameter Control

Most operators of metaheuristics have parameters, and their values usually have impact on the algorithm performance. Besides tuning parameter values manually, some algorithms have dynamic or adaptive control mechanisms. Pandit *et al*. [19] solved the DEED problem by an improved bacteria foraging algorithm (IBFA), which searches the solution space by imitating the movement of bacteria. The step size of movement was controlled by a logistic mapping function. In MOHDE-SAT [14], the authors set the scaling factor (F) of the mutation operator of DE by values that decrease exponentially as the generation number increases. The authors of [13] adopted a similar strategy to control both F and the crossover rate (CR). Li *et al*. [20] used the harmony search (HS), which mimics the improvisation process of musicians. The bandwidth of HS plays the role of step size of movement in the search space. Li *et al*. divided the search process into three stages and set different and static values to the bandwidth in these stages. The values are smaller for later stages. In EFDE [10], the authors set F of DE mutation operator by random values every time when mutation was carried out. As for the values of CR of crossover, they were set by a weighted sum of the CR value of the best solution in the population and a prefixed constant value. If the parent solution has worse fitness, it is more likely that the weight of the CR value of the best solution is higher.

III. PROPOSED ALGORITHM

In this paper, we solve the DEED problem by an algorithm based on MOEA/D-DRA [21], which is a variant of MOEA/D. To fit the constraints of DEED, we add a repair mechanism to ensure that the solutions are feasible. Moreover, we use multiple mutation operators and adopt an adaptive control mechanism. We call our algorithm MOEA/D-DRAM (MOEA/D with dynamic resource allocation and mutation). Algorithm 1 shows the main flow. Details are presented in the following subsections.

Algorithm 1 MOEA/D-DRAM

NP: the number of sub-problems, also the population size
π: utility of sub-problems, initially an NP-dimension vector with all 1
$B(i)$: neighborhood of sub-problem i
λ: objective weight vectors of sub-problems
M: mating pool
pr: probability of selection of mutation strategies, initially [0.5, 0.5]
H_{res}: history for dynamic resource allocation
H_{mut}: history for adaptive selection of mutation strategies

```
pop = Initialize(NP)
for i = 1, …, NP do
  pop[i] = Repair(pop[i])
end for
z* = GetReferencePoint(pop)
g = 1
while the stopping criterion is not met do
  for j = 1, …, NP do
    s = SelectSubproblem (π)
    if rand(0, 1) < δ do
      M = B(s)
    else
      M = {1, 2, …, NP}
    end if
    k = SelectMutation(pr)
    u′ = Reproduce(pop, s, M, mutation[k])
    u′ = Repair(u′)
    pop = Replace(u′, pop, M, λ, z*, n_r)
    H_mut = Record(H_mut, k, pop[s], u′)
  end for
  z* = GetReferencePoint(pop)
  pr = UpdateProbability(H_mut)
  H_res = Record(H_res, pop, g)
  if g mod θ == 0
    π = UpdateUtility(H_res)
  end if
  g = g + 1
end while
return pop
```

A. Encoding and Initialization

The DEED problem aims to decide the power output of N generation units in the power system over T periods to satisfy constraints and minimize cost and emissions. It is natural to encode a solution x_j (j = 1, 2, …, NP) as a real vector $[x_{j,1,1}, x_{j,2,1}, \dots, x_{j,N,T}]$. For example, a solution is a 120-D real vector if there are five generation units and the number of periods is 24. Each solution in the initial population is generated randomly by (7), where P_i^{max} and P_i^{min} are the power limit of unit i.

$$x_{j,i,t} = P_i^{min} + rand(0,1) \times \left(P_i^{max} - P_i^{min}\right), i = 1 \dots N, t = 1 \dots T \tag{7}$$

B. Repair Mechanism

As mentioned in (3)–(6), the DEED problem has three types of constraints. Solutions generated during initialization and reproduction might not satisfy these constraints. Thus, we apply a repair procedure to ensure their feasibility. The steps of repairing an infeasible solution x_j are as follows:

Step 0. Set time period t = 1 and number of trials k = 0.

Step 1. Calculate the lower bound and upper bound values of $x_{j,i,t}$ (i = 1, 2, …, N) by considering the power limit constraints and ramp rate limit constraints.

Step 2. Check the values of all $x_{j,i,t}$. If the power output of a unit is not in the feasible range, set it to the closest boundary value.

Step 3. Deal with the power balance constraint:

Step 3.1. If the power balance constraint is satisfied, go to Step 4.

Step 3.2. If k >= 100, go to Step 5; otherwise, deal with the power balance constraint. (Details are given later.), $k = k + 1$, and go back to Step 1.

Step 4. If t is smaller than T, $t = t + 1$ and k = 0. Go back to Sep 1. Otherwise, go to Step 5.

Step 5. If the solution is feasible, stop; otherwise, generate a new random solution and go back to Step 0. (In our practice, the success rate of repairing is almost 100%, and thus infinite looping does not happen.)

In Step 3.2, we need a method to meet the power balance constraint. When the constraint is not satisfied, it means there is a non-zero difference $V_{j,t}$ between the demand and the generated power plus loss, i.e. $V_{j,t} = P_{D,t} - (\sum_{i=1}^{N} x_{j,i,t} + L(x_j))$. (In practice, we regard a solution as feasible when the absolute value of $V_{j,t}$ is smaller than 10^{-5} in Step 3.1.) To meet the constraint, a simple method is to adjust the power output of a single unit by adding $V_{j,t}$. Another method is to average the difference $V_{j,t}$ over all units and add the average value to all units. Here, we adopt an extended version called proportional dynamic adjustment decision variable method (PDAD) from [13]. As the name reveals, the extended version considers the adjustable range of power output of generation units and distributes the difference to all units proportionally. Technically speaking, the adjustment is calculated according to (8) and (9).

$$R_i = \left(P_i^{max} - P_i^{min}\right) / \sum_{k=1}^{N} \left(P_k^{max} - P_k^{min}\right) \tag{8}$$

$$x_{j,i,t} = x_{j,i,t} + V_{j,t} \cdot R_i \tag{9}$$

C. Neighborhood and Mating Selection

The core idea of MOEA/D is to solve a multiobjective optimization problem by decomposing the problem into many single-objective sub-problems. The objective function of each sub-problem is an aggregated form of the original objectives, such as weighted summation or weighted Tchebycheff functions. Sub-problems are distinguished by the weights of objectives; for example, one sub-problem aims to minimize $0.3 \cdot f_1 + 0.7 \cdot f_2$ and another aims to minimize $0.6 \cdot f_1 + 0.4 \cdot f_2$. The DEED problem considers two objectives, and hence the objective weight vector has two dimensions. Let NP denote the number of sub-problems, we set the objective weight vector λ_j of the j^{th} (j = 1, 2, …, NP) sub-problem by $\lambda_j = [(j-1)/(NP-1), 1-(j-1)/(NP-1)]$, which aims to cover the objective space with even distribution.

Distance between sub-problems j and j' is defined by the Euclidean distance between their objective weight vectors λ_j and $\lambda_{j'}$. The neighborhood $B(j)$ of a sub-problem j is the set of N_{nb} sub-problems that have the smallest distance to j. When we are producing a new solution for sub-problem j, only solutions of sub-problems in $B(j)$ can participate. This realizes a concept called mating restriction. In MOEA/D, the mating pool of a sub-problem is defined as its neighborhood in probability δ and as the whole population in probability $(1 - \delta)$. The values of N_{nb} and δ will be given in Section IV.

D. Mutation and Crossover

We produce new solutions by classical DE operators. The rand/1 and best/1 mutation operators are presented in (10) and (11), respectively. The new solution v_j produced by either (10) or (11) is called the mutant vector. The parameter F is called the scaling factor. Solutions x_{r1}, x_{r2}, and x_{r3} are randomly selected from the mating pool. Among all solutions except the current solution of the sub-problem, the solution with the best aggregated objective value is selected as x_{best}.

$$\boldsymbol{v}_{j,i,t} = \boldsymbol{x}_{r_1,i,t} + F \cdot \left(\boldsymbol{x}_{r_2,i,t} - \boldsymbol{x}_{r_3,i,t}\right) \tag{10}$$

$$\boldsymbol{v}_{j,i,t} = \boldsymbol{x}_{best,i,t} + F \cdot \left(\boldsymbol{x}_{r_1,i,t} - \boldsymbol{x}_{r_2,,i,t}\right) \tag{11}$$

The mutation operator produces a mutant vector v_j, and the crossover operator takes v_j and x_j (current solution of sub-problem j) to produce a trial vector u_j. Here we use the binomial crossover, as (12) shows. The parameter CR refers to the crossover rate.

$$u_{j,i,t} = \begin{cases} v_{j,i,t} & if\ rand(0,1) < CR \\ x_{j,i,t} & otherwise \end{cases} \tag{12}$$

To increase population diversity, we further apply the polynomial mutation [22] to the trial vector, as shown in (13) and (14). The variable r_i has a random value in the interval [0, 1], and the value of parameter p_m will be given in Section IV.

$$u'_{j,i,t} = u_{j,i,t} + \left(P_i^{max} - P_i^{min}\right) \times \delta, \qquad \text{if } rand(0,1) < p_m \tag{13}$$

$$\delta = \begin{cases} (2r_i)^{1/(\eta+1)} - 1 & \text{if } rand(0,1) < 0.5 \\ 1 - \left(2(1-r_i)\right)^{1/(\eta+1)} & otherwise \end{cases} \tag{14}$$

E. Fitness Assignment and Environmental Selection

We use the weighted Tchebycheff function to define the objective function of each sub-problem, as shown in (15). In (15), x refers to a solution, $f'_m(\boldsymbol{x})$ refers to the m^{th} normalized objective value (cost or emissions in DEED), and $\boldsymbol{\lambda_j} = (\lambda_{j,1}, \lambda_{j,2})$ refers to the objective weight vector. The objective values are normalized since cost and emissions may fall in very different ranges. The reference point $\mathbf{z}^*$ is defined by $\mathbf{z}^* = (z_1^*, z_2^*)$, where z_k^* is the minimal value of the m^{th} normalized objective function over all solutions in the population. In the normalized 2-D objective space, it is actually (0, 0).

$$g^{te}\left(\boldsymbol{x}|\boldsymbol{\lambda_j}, \boldsymbol{z}^*\right) = \max_{m=1,2}\left\{\lambda_{j,m} \cdot |f'_m(\boldsymbol{x}) - z_m^*|\right\}, \qquad j = 1, \ldots, NP \tag{15}$$

To solve a sub-problem j, we apply three operators mentioned in the previous sub-section to produce a new solution u'. The solution u' may replace only the solutions of the mating pool of sub-problem j. If $g^{te}(\boldsymbol{u'}|\boldsymbol{\lambda_k}, \boldsymbol{z}^*)$ is smaller than $g^{te}(\boldsymbol{x_k}|\boldsymbol{\lambda_k}, \boldsymbol{z}^*)$, which means that u' has better solution quality than x_k in terms of the aggregated objective function of sub-problem k, the solution u' will replace solution x_k. The solutions of sub-problems in the neighborhood are randomly selected and compared. To avoid losing diversity, at most n_r solutions can be replaced by one new solution. The value of n_r will be given in Section IV.

F. Dynamic Resource Allocation (DRA)

In MOEA/D, each sub-problem is solved once in each generation. However, some sub-problems might be easier to solve than others, and the computational resource could be saved to solve other sub-problems. The core concept of the DRA mechanism of MOEA/D-DRA is to measure the possibility of improving the solution quality (called utility) of sub-problems and allocate more computational resource to the sub-problems with higher utility. The utility π_j of sub-problem j is defined by (16) and (17). Every θ generations, the utility of each sub-problem is measured. Let x^{old} denote the solution of sub-problem at generation ($g - \theta$) and x^{new} denote the solution at current generation g. The improvement percentage Δ is calculated by (17). If Δ is greater than 0.001, the utility of the sub-problem is set by one; otherwise, it decreases a little bit by (16). In one generation, NP sub-problems are selected to be solved by 10-tournament in terms of the utility, where NP is the number of sub-problems (and is also the population size).

$$\pi_j = \begin{cases} 1 & \text{if } \Delta_j > 0.001 \\ \left(0.95 + 0.05 \cdot \dfrac{\Delta_j}{0.001}\right) \cdot \pi_j & \text{otherwise} \end{cases} \tag{16}$$

$$\Delta_j = \frac{g^{te}\left(\mathbf{x}_j^{\boldsymbol{old}}|\boldsymbol{\lambda_j}, \boldsymbol{z}^*\right) - g^{te}\left(\mathbf{x}_j^{\boldsymbol{new}}|\boldsymbol{\lambda_j}, \boldsymbol{z}^*\right)}{g^{te}\left(\mathbf{x}_j^{\boldsymbol{old}}|\boldsymbol{\lambda_j}, \boldsymbol{z}^*\right)} \tag{17}$$

G. Dynamic Selection of Mutation Operators

As mentioned, we use two DE mutation operators, rand/1 and best/1. The rand/1 mutation has better exploration ability, whereas the best/1 mutation has better exploitation ability. To utilize them appropriately, we adopt an adaptive operator selection mechanism.

At the end of the g^{th} generation, we calculate the credit of each mutation operator. Let $S_k = \{s_{k,1}, s_{k,2}, \ldots\}$ denote the set of sub-problems that use the k^{th} mutation operator to produce the new solution. The credit $c_{k,g}$ of the k^{th} mutation operator at the g^{th} generation is defined by (18). The accumulated credit $q_{k,g}$ is updated by (19), where α is a learning rate.

$$c_{k,g} = \sum_{j \epsilon S_k} \max\left\{0, \frac{g^{te}(x_j, \boldsymbol{\lambda_j}, \boldsymbol{z}^*) - g^{te}(u'_j, \boldsymbol{\lambda_j}, \boldsymbol{z}^*)}{g^{te}(x_j, \boldsymbol{\lambda_j}, \boldsymbol{z}^*)}\right\} \tag{18}$$

$$q_{k,g+1} = (1-\alpha) \times q_{k,g} + \alpha \times c_{k,g} \tag{19}$$

Based on the accumulated credits of mutation operators, we use the probability matching method to decide the probability of selecting these operators. The probability $pr_{k,g+1}$ of selecting the k^{th} mutation operator at the $(g+1)$ generation is defined in (20). The parameter $pr_{\min}$ is defined to keep the minimal probability of selecting each mutation operator. K refers to the number of mutation operators and is two in our current algorithm.

$$pr_{k,g+1} = pr_{min} + (1 - K \times pr_{min}) \times \frac{q_{k,g+1}}{\sum_{l=1}^{K} q_{l,g+1}} \tag{20}$$

IV. Experiments and Results

A. Test Cases

We used six test cases to examine the performance of our algorithm and its components. Model coefficients of these test cases are public and available from the literature. One of the six models does not include the valve point effect (i.e. d_i is zero in the cost function (1)), and two of them do not consider loss (i.e. $L(P_t)$ is zero in the power balance constraint (4)). In all test cases, the number of time periods (T) is 24. Table I summarizes these test cases.

TABLE I. Test Cases

Test case	Data source	Number of units (N)	Valve point effect?	Loss considered?
5-U	[1]	5	Y	Y
6-U	[8]	6	N	Y
10-U	[2]	10	Y	Y
15-U	[7]	15	Y	Y
30-U	[9]	30	Y	N
40-U	[13]	40	Y	N

B. Performance Indicators

We solve the DEED problem by a Pareto-based approach. It means that we aim to find the approximation set of the Pareto optimal solutions. To evaluate a set of solutions, we adopt a popular performance indicator in the field of multiobjective optimization, the inverted generational distance (IGD). It calculates the average distance from the reference front to the approximation front, as shown in (21).

$$IGD(R, F) = \frac{\sum_{x \in R} d(x, F)}{|R|} \tag{21}$$

The set F denotes the approximation front obtained by an algorithm, and the set R denotes the reference set. Typically, the reference set is formed by sampling solutions from the true Pareto optimal set. However, the true Pareto optimal front is not available for the DEED test cases. In our experiments, we ran our MOEA/D-DRAM with one million fitness function evaluations (simply speaking, one million solutions were produced and evaluated) for twenty times and then collected the net set of non-dominated solutions as R. When $IGD(R, F_1)$ is smaller than $IGD(R, F_2)$, we say the approximation set F_1 is better than F_2.

When we compared the performance of algorithm variants, we ran each algorithm variant to solve each test case for twenty times and obtain twenty approximation sets. The IGD value of each approximation set was calculated. The average and standard deviation over twenty IGD values are reported in the tables in the sub-sections D–G. We also did the Wilcoxon rank-sum test with the level of significance 0.05 to check whether the IGD values of the tested algorithms are significantly different. In tables, the symbol + means that our algorithm significantly outperforms others. The symbols = and − refer to the cases of equal and worse performance, respectively.

Since most existing studies only provided the extreme solutions and/or the best compromise solution, we cannot compare our algorithm with them in terms of IGD. Therefore, in the last sub-section, we compared algorithms by the extreme solutions and the best compromise solution. The extreme solutions refer to the solutions with the minimal cost or the minimal emissions. As for the best compromise solution, we select the solution with the largest μ_j defined in (22). In (23), f_m^{min} and f_m^{max} refer to the minimal and maximal value of the m^{th} objective function over all solutions in the approximation set, respectively.

$$\mu_j = \frac{\sum_{m=1}^{2} \mu_{j,m}}{\sum_{j=1}^{NP} \sum_{m=1}^{2} \mu_{j,m}} \tag{22}$$

$$\mu_{j,m} = \begin{cases} 1 & f_m(\mathbf{x}_j) \le f_m^{min} \\ \dfrac{f_m^{max} - f_m(\mathbf{x}_j)}{f_m^{max} - f_m^{min}} & f_m^{min} < f_m(\mathbf{x}_j) < f_m^{max} \\ 0 & f_m(\mathbf{x}_j) \ge f_m^{max} \end{cases} \tag{23}$$

C. Parameter Setting

Our algorithm adopts several mechanisms and has several parameters. Table II summarizes the parameters and their purposes. For the four parameters related to reproduction operators, we set them by typical setting in the literature. For the four parameters of MOEA/D and DRA, we set them by the setting in the paper of MOEA/D-DRA [21]. As for other parameters, we did some preliminary tests and set their values accordingly.

TABLE II. PARAMETERS AND VALUES

Origin	Parameter	Purpose	Value
EA	*MaxFFE*	maximum number of FFE	100000
	NP	population size	100
MOEA/D	δ	prob. of mating restriction	0.9
	N_{nb}	neighborhood size	10
	n_r	number of replaced solutions	1
DRA	θ	learning period	10
Reproduction operator	F	scaling factor of mutation	0.5
	CR	crossover rate	0.5
	p_m	rate of polynomial mutation	1/N
	η	range of polynomial mutation	20
Adaptive mutation	α	learning rate	0.5
	pr_{min}	min. prob. of each mutation	0.1

D. Effects of the Repair Mechanism

There are three types of constraints in DEED, and thus constraint handling is an important task in solving DEED. In this experiment, we tested three repair mechanisms and compared their performance in terms of (1) successful repair rate and (2) the IGD of solutions obtained by MOEA/D taking each of them as the repair procedure. The three repair mechanisms are single-repair (SR) mechanism (adjusting one unit at a time), all-repair (AR) mechanism (adjusting all units), and PDAD mechanism (adjusting based on range of values). Details of them are described in section III-B.

First, we applied each repair mechanism to repair 5000 random solutions and counted the number of successful trials. Fig. 1 shows the successful rate under maximal number of repairing iterations 10, 25, and 50. If the solution satisfies all constraints within the maximal number of iterations, that trial is regarded as a success. The x-axis refers to the test cases, and the y-axis refers to the successful rate.

When the maximal number of iterations is 10, the SR mechanism has higher successful rates than the two others. When the number of iterations increases to 25, the PDAD mechanism has the highest successful rates. Given 50 iterations, the PDAD can achieve almost 100% successful rates for all six test cases. In general, it is harder to repair when the problem scale gets larger. The 15-U test case is particularly hard for AR and PDAD.

Next, we adopted each of the three mechanisms in the repair step in MOEA/D. The maximal number of iterations was set by 50. Here we used the pure MOEA/D. The DRA mechanism and dynamic mutation mechanism were not included. Solutions were produced by the DE/rand/1/bin strategy and polynomial mutation. Table III presents the IGD results. It is clear that PDAD can help MOEA/D to obtain the best approximation set of solutions. The algorithm variant using PDAD significantly outperforms the variant using SR in solving five test cases and outperforms the variant using AR in solving four test cases, respectively. The reason could be that SR and AR have lower successful repair rates and consume fitness function evaluations without generating useful solutions. Since the ranges of power limits of all units in the test case 6-U are the same, the two algorithm variants using AR and PDAD have the same results. In the following experiments, we adopted PDAD as the repair mechanism and set the maximal number of repairing iterations by 100 to keep high sucessful repair rates.

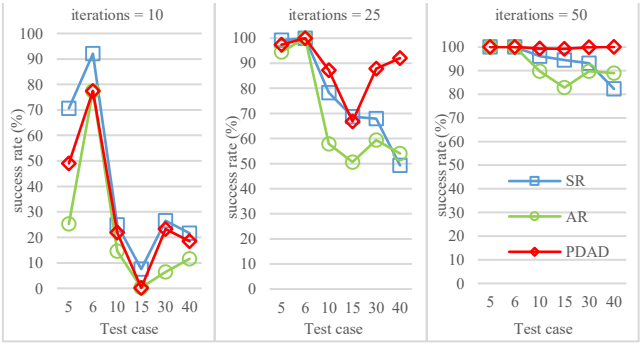


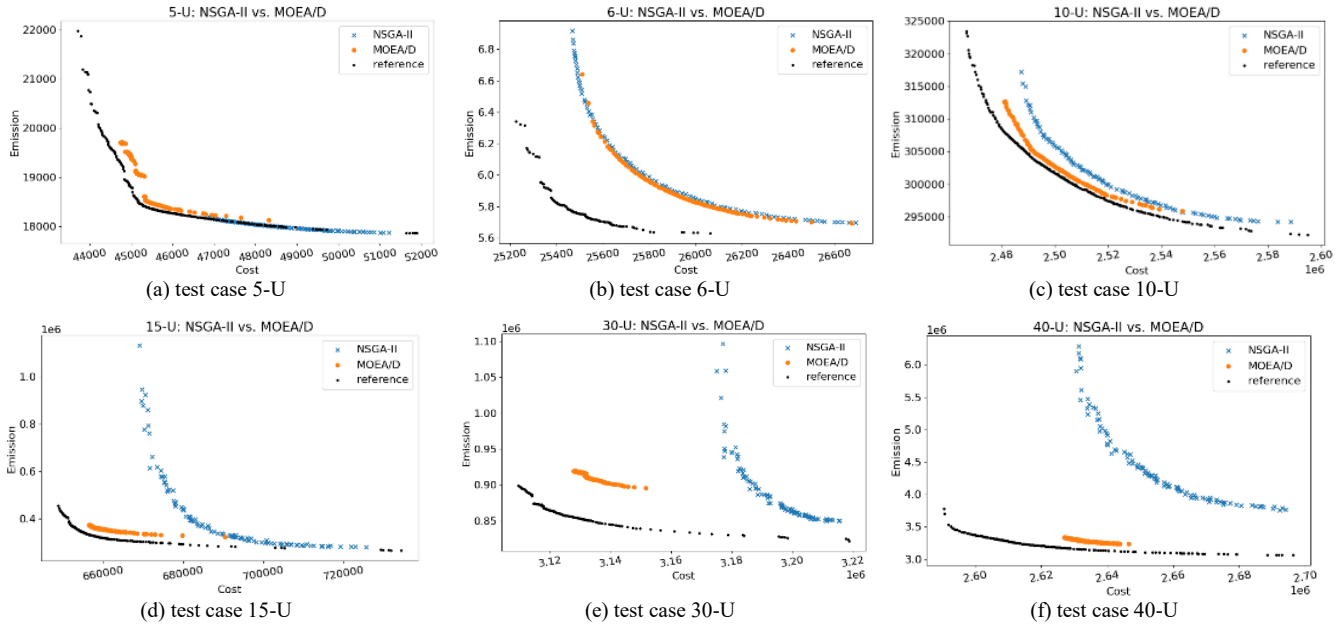
Fig. 1. The successful rates of three repair mechanisms for six test cases

TABLE III. IGD COMPARISON OF MOEA/D USING THREE REPAIR MECHANISMS

	SR			AR			PDAD	
Test case	Avg.	Std.		Avg.	Std.		Avg.	Std.
5-U	0.07877	0.01366	=	0.08470	0.01726	+	**0.07284**	0.01330
6-U	0.33790	0.01532	+	**0.26933**	0.01232	=	**0.26933**	0.01232
10-U	0.11101	0.00565	+	0.06419	0.00501	+	**0.04707**	0.00534
15-U	0.28399	0.03952	+	0.16277	0.01230	+	**0.14364**	0.01145
30-U	0.51606	0.02067	+	0.47529	0.02490	=	**0.46785**	0.02848
40-U	0.61316	0.02623	+	0.26296	0.00870	+	**0.23878**	0.00686
+/=/–	5/1/0			4/2/0				

E. Effects of the Multiobjective Optimization Mechanism

In this experiment, we compared performance of the decomposition-based approach (MOEA/D) and the dominance-based approach (NSGA-II) in solving the DEED problems. Table IV presents the IGD values of the two algorithm variants based on NSGA-II and MOEA/D. MOEA/D significantly outperforms NSGA-II in solving five out of six test cases. Fig. 2 shows the approximation sets of solutions obtained by the two compared algorithms and the reference sets of solutions. We can find that NSGA-II gets stuck during moving toward the reference front and spreads the front in an area still far away from the reference front. In most cases, MOEA/D keeps better converging ability and hence has better solution quality. Performance difference gets more obvious as the problem scale gets larger. One exception is the test case 6-U, where MOEA/D also gets stuck and NSGA-II has a better spread.



(a) test case 5-U (b) test case 6-U (c) test case 10-U

(d) test case 15-U (e) test case 30-U (f) test case 40-U

Fig. 2. Solutions obtained by NSGA-II and MOEA/D for six test cases

TABLE IV. IGD COMPARISON OF TWO TYPES OF MO SELECTION

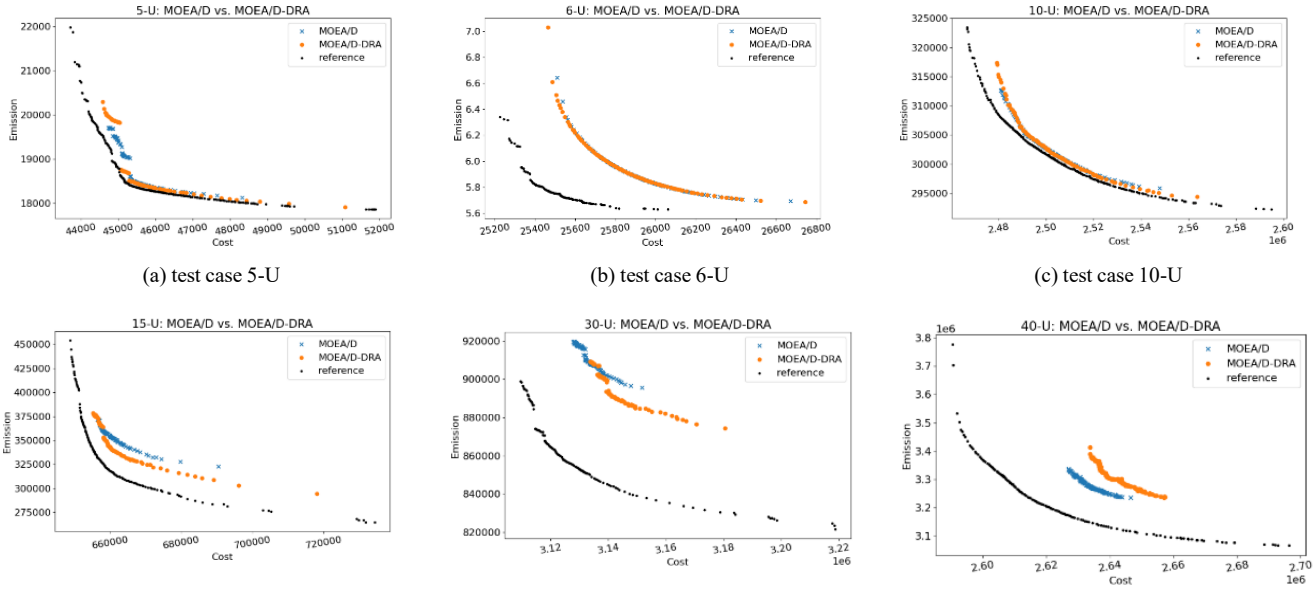
	NSGA-II			MOEA/D	
Test case	Avg.	Std.		Avg.	Std.
5-U	0.20054	0.01068	+	**0.07284**	0.01330
6-U	**0.24886**	0.00494	–	0.26933	0.01232
10-U	0.07616	0.00458	+	**0.04747**	0.00691
15-U	0.25066	0.02506	+	**0.14197**	0.01392
30-U	0.53718	0.03656	+	**0.47031**	0.02811
40-U	0.42234	0.02257	+	**0.23878**	0.00686
+/=/–		5/0/1			

F. *Effects of Dynamic Resource Allocation*

After confirming the positive effect of MOEA/D, in this experiment we examine the effect of the DRA mechanism (Section III-F). Table V presents the IGD values of the two algorithm variants using MOEA/D and MOEA/D-DRA. MOEA/D-DRA significantly outperforms MOEA/D in solving five out of six test cases. The core idea of DRA is to distribute computational resource to sub-problems based on the search progress of them. Here we take the test case 10-U as an example. We counted the number of times that each sub-problem was solved over twenty runs and show the results in Fig. 4(a). Fig. 4(a) shows that MOEA/D-DRA solved sub-problems with low and high weight values of the cost objective by more times. Solving these sub-problems corresponds to searching for solutions around the two ends of the front. More computational resource on these sub-problems helps to find more solutions around the two ends, as shown in Fig. 3(c).

TABLE V. IGD COMPARISON OF MOEA/D AND MOEA/D-DRA

	MOEA/D			MOEA/D-DRA	
Test case	Avg.	Std.		Avg.	Std.
5-U	0.07284	0.01330	+	**0.05902**	0.01963
6-U	0.26933	0.01232	+	**0.23243**	0.00290
10-U	0.04747	0.00691	+	**0.03985**	0.00440
15-U	0.14197	0.01392	+	**0.12597**	0.01206
30-U	0.47031	0.02811	+	**0.40611**	0.03194
40-U	**0.23878**	0.00686	=	0.24149	0.01253
+/=/–		5/0/1			



(a) test case 5-U (b) test case 6-U (c) test case 10-U

(d) test case 15-U (e) test case 30-U (f) test case 40-U

Fig. 3. Solutions obtained by MOEA/D and MOEA/D-DRA for six test cases

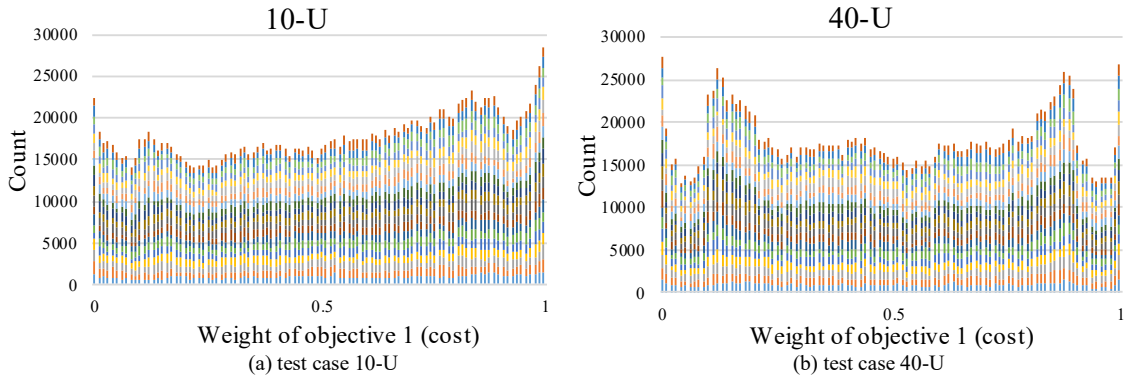


Fig. 4. The number of times sub-problems are solved over twenty runs

The DRA mechanism does not always lead to better performance. In Fig. 4(b), we see that DRA solved sub-problems that have the weight value of the cost objective in the interval [0.3, 0.7] by fewer times. These sub-problems correspond to the middle part of the front. The insufficient computational resource on these sub-problems causes that MOEA/D-DRA does not converge toward the reference front as well as MOEA/D, as shown in Fig. 3(f).

G. Effects of Mutation Operators and Dynamic Selection

In this experiment, we examine the effects of mutation operators and dynamic selection of operators. We tested four algorithm variants. Two variants used rand/1 and best/1 mutation operators, respectively; the third variant selected one operator between rand/1 and best/1 randomly; the last variant used our dynamic selection mechanism (Section III-G). Table VI presents the IGD values of these four algorithm variants.

TABLE VI. IGD COMPARISON OF MOEA/D-DRA USING DIFFERENT MUTATION STRATEGIES

Test case		rand/1		best/1		random		dynamic
5-U	Avg.	0.05902	+	0.05613	+	0.05432	+	**0.04469**
	Std.	0.01963		0.01567		0.01717		0.01217
6-U	Avg.	0.23243	+	0.22151	−	**0.21751**	−	0.22574
	Std.	0.00290		0.00199		0.00156		0.00363
10-U	Avg.	0.03985	=	0.04751	+	0.04133	+	**0.03747**
	Std.	0.00440		0.00419		0.00450		0.00367
15-U	Avg.	0.12597	=	0.13387	=	0.12364	=	**0.12181**
	Std.	0.01206		0.02435		0.01338		0.01043
30-U	Avg.	0.40611	+	**0.29664**	−	0.35208	+	0.32804
	Std.	0.03194		0.04490		0.03171		0.02901
40-U	Avg.	**0.24149**	−	0.26197	=	0.30669	+	0.28199
	Std.	0.01253		0.01931		0.01771		0.02233
+/=/–		3/1/2		2/2/2		4/1/1		

First, we can see that the algorithm variant using our dynamic selection mechanism performs the best in solving three out of six test cases. In addition, this variant does not perform the worst in any test case. Each of the other three variants performs the best in one test case. Using multiple operators either randomly or adaptively is helpful for solving the four small- or medium-scale test cases. For the test case 30-U, the variant using best/1 mutation performs much better than the variant using rand/1 mutation. In this case, adding rand/1 as the second way of producing new solutions could be useless. Another possible reason for the ineffectiveness of dynamic selection in solving 30-U and 40-U test cases might be that the search space is very large and more computational resource is required to identify which mutation operator is more effective to generate good solutions.

H. Comparison with Existing Studies

In the final part of this section, we compare our MOEA/D-DRAM algorithm with those in the literature. We included 15 papers published during 2008–2021, as summarized in Table VII. We classify them into SO and MO approaches. Papers that aimed to find a single optimal solution are classified as SO approaches, and papers that aimed to find a set of non-dominated solutions in every single run are classified as MO approaches.

Here we only focused on three test cases 5-U, 6-U, and 10-U since they were more popular in the literature. (The number of decision variables of these test cases are 120, 144, and 240, respectively, due to 24 time periods. They are not easy problems as they may look like.) To compare performance of metaheuristics, the number of fitness function evaluations (FFE) is a quite important factor. In the literature on DEED, however, there was no consistent setting of FFE for solving these test cases. Therefore, we estimated the FFE by the population size and the generation number in the literature and listed the FFE in Table VII. A cell "n/a" means that we did not find information about the FFE. A blank cell means that the objective values of solutions for the test case were not provided. Among 15 existing papers, only one paper provided solutions of all three test cases.

TABLE VII. INFORMATION ON BENCHMARK ALGORITHMS

Algorithm	Publication Year	Type	Run	Estimated FFE ($\times 10^4$)		
				5-U	6-U	10-U
NSGA-II [2]	2008	MO	20			0.2
IBFA [19]	2012	SO	25			8
GSOMP [8]	2012	MO	50		6	
DE-SQP [12]	2013	SO	30	120		120
MAMODE [5]	2013	MO	n/a		60	20
MOHDE-SAT [14]	2015	MO	10	5		5
MNSGA-II [15]	2016	MO	n/a			2
HCRO [9]	2016	MO	30			5
PSOAWL [6]	2017	SO	10			10
MONNDE [7]	2018	SO	10			n/a
IMOEA/D-CH [17]	2019	MO	30		20	20
EFDE [10]	2019	SO	20	10		40
NEHS [20]	2019	SO	50	5	5	10
PSOCS [11]	2020	SO	25	6		6
ITSA [18]	2021	SO	30	8		8
MOEA/D-DRAM		MO	20	5	5	5/10/20

We compare our MOEA/D-DRAM with six algorithms in solving the 5-U test case. The IGD values are presented in Table VIII, and the solutions are plotted in Fig. 5. Five benchmark algorithms are SO approaches. DE-SQP and PSOCS found solutions with much smaller cost than other algorithms did, but it should be noted that DE-SQP consumed about twenty times of FFE. The best compromise solutions of ITSA, EFDE, and NEHS have similar objective values as that of MOEA/D-DRAM does. However, MOEA/D-DRAM is able to find the a whole set of non-dominated solutions in a single run. MOHDE-SAT is the only benchmark algorithm of MO type. Although MOHDE-SAT found the best-emission solution with slightly lower emissions than that of MOEA/D-DRAM, it is not able to find solutions with low cost. In Fig. 5, we can see that all three solutions of MOHDE-SAT are near the bottom right part of the approximation front of MOEA/D-DRAM. MOHDE-SAT is based on NSGA-II. Its failure of finding low-cost solutions is consistent with what we found in our experiments on comparing NSGA-II and MOEA/D in Section IV-E (see Fig. 2(a)).

Four existing algorithms were compared in solving the test case 6-U. MAMODE and GSOMP are MO approaches, and their best compromise solutions are dominated by the solutions of MOEA/D-DRAM. NEHS is an SO approach. It consumed the same FFE as that of MOEA/D-DRAM to find a single solution that has similar objective values as those of some solutions in the approximation front of MOEA/D-DRAM. Checking the three solutions of IMOEA/D-CH, the shapes of front of IMOEA/D-CH and MOEA/D-DRAM look similar. The solutions of IMOEA/D-CH have better objective values than those of MOEA/D-DRAM do, but IMOEA/D-CH consumed four times of FFE of MOEA/D-DRAM.

TABLE VIII. SOLUTIONS OF TEST CASE 5-U

Algorithm (FFE: $\times 10^4$)	Type	Objective	Best Cost	Best Emission	Best Compromise
MOHDE-SAT (5)	MO	f_{cost}	46478	50681	48214
		$f_{emission}$	18208	**17884**	18011
PSOCS (6)	SO	f_{cost}			43329
		$f_{emission}$			19934
ITSA (8)	SO	f_{cost}			45971
		$f_{emission}$			18370
EFDE (10)	SO	f_{cost}			45242
		$f_{emission}$			18417
NEHS (5)	SO	f_{cost}			45398
		$f_{emission}$			18392
DE-SQP (120)	SO	f_{cost}	**43161**		44450
		$f_{emission}$	23080		19616
MOEA/D-DRAM (5)	MO	f_{cost}	44133.7	51613.1	45523.4
		$f_{emission}$	22258.1	17888	18438

TABLE IX. SOLUTIONS OF TEST CASE 6-U

Algorithm (FFE: $\times 10^4$)	Type	Objective	Best Cost	Best Emission	Best Compromise
MAMODE (60)	MO	f_{cost}			25912
		$f_{emission}$			5.9795
IMOEA/D-CH (20)	MO	f_{cost}	**25367**	**26679**	**25676**
		$f_{emission}$	**6.9218**	**5.6677**	**5.9720**
NEHS (5)	SO	f_{cost}			26295
		$f_{emission}$			5.7276
GSOMP (6)	MO	f_{cost}			25924
		$f_{emission}$			6.0041
MOEA/D-DRAM (5)	MO	f_{cost}	25461.2	26724.5	25750.8
		$f_{emission}$	6.9962	5.6878	6.0083

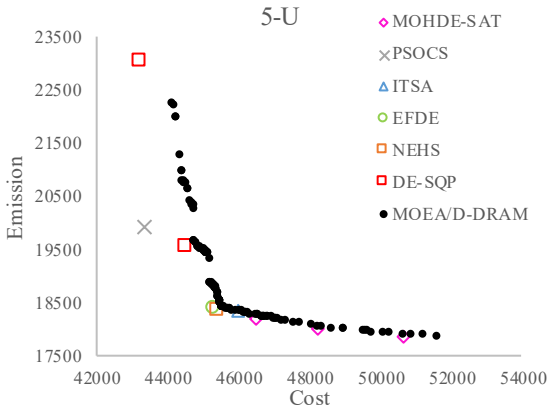


Fig. 5. Solutions of test case 5-U

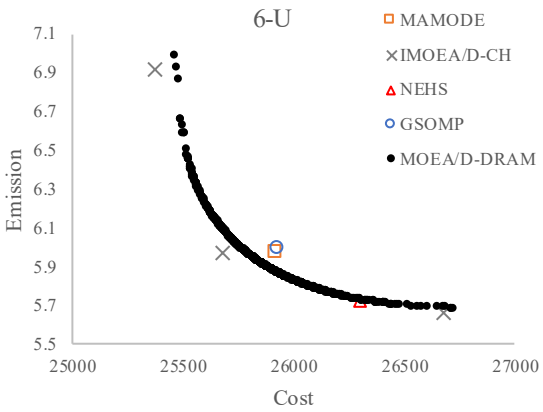


Fig. 6. Solutions of test case 6-U

The test case 10-U was widely studied in the literature. We compared MOEA/D-DRAM with 14 algorithms. Since these algorithms consumed very different number of FFE, we separated them into three groups based on FFE: a group using 50000 or fewer FFE, a group using 60000 to 100000 FFE, and a group using 200000 or more FFE. To compare with these three groups of algorithms, we ran MOEA/D-DRAM using 50000, 100000, and 200000 FFE, respectively.

Table X and Fig. 7 show the results of the group of algorithms using 50000 or fewer FFE. All compared algorithms are MO approaches. The best compromise solutions of NSGA-II and MNSGA-II are obviously worse than the solutions of other algorithms. This might be due to that they consumed fewer FFE. The other three algorithms consumed the same FFE. The results showed that MOEA/D-DRAM not only found solutions of the lowest cost and the lowest emissions, it also found the approximation front with much wider spread than MOHDE-SAT and HCRO did.

TABLE X. SOLUTION OF TEST CASE 10-U USING 50000 OR FEWER FFE

Algorithm (FFE: $\times 10^4$)	Type	Objective	Best Cost	Best Emission	Best Compromise
NSGA-II (0.2)	MO	f_{cost}			2.5226
		$f_{emission}$			3.0994
MNSGA-II (2)	MO	f_{cost}			2.5552
		$f_{emission}$			2.9914
MOHDE-SAT (5)	MO	f_{cost}	2.5082	2.5470	2.5279
		$f_{emission}$	3.0146	2.9607	2.9776
HCRO (5)	MO	f_{cost}	2.4799	2.5201	2.5171
		$f_{emission}$	3.2135	2.9846	2.9907
MOEA/D-DRAM (5)	MO	f_{cost}	**2.4796**	2.5887	2.5054
		$f_{emission}$	3.1948	**2.9401**	3.0323

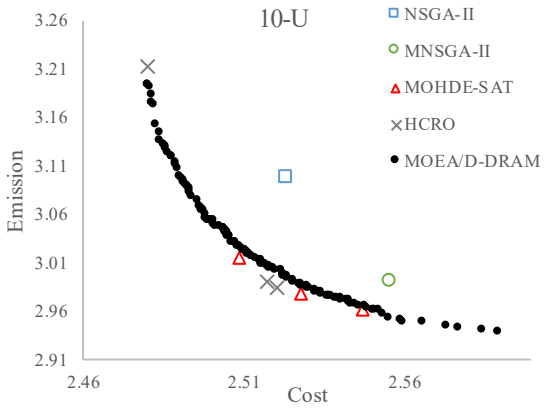


Fig. 7. Solutions of test case 10 using 50000 or fewer FFE

Table XI and Fig. 8 show the results of the group using 60000 to 100000 FFE. Note that in this group, only MOEA/D-DRAM is an MO approach. By spending all FFE on searching for the single best compromise solution, ITSA and NEHS can find solutions with better objective values than those of MOEA/D-DRAM. Solutions of PSOCS and PSOAWL have similar objective values as those of MOEA/D-DRAM. The best compromise solution of IBFA has similar objective values as those of MOEA/D-DRAM, but the extreme solutions of IBFA were dominated by solutions of MOEA/D-DRAM.

The results of the last group using 200000 or more FFE are shown in Table XII and Fig. 9. Our MOEA/D-DRAM found the best-emission solution with the lowest emissions among the three MO algorithms. It also found the best-cost solution with the second lowest cost among all algorithms. The cost of the best-cost solution is only slightly higher than that of the best-cost solution of DE-SQP, which consumed six times of FFE as that of MOEA/D-DRAM. DE-SQP can find solutions of low cost, but its best compromise solution is biased toward cost, not really striking a balance between cost and emissions. The best compromise solution of EFDE has better objective values than those of solutions of MOEA/D-DRAM. However, EFDE, as an SO approach, can only find a single solution at a time and took twice FFE as that of MOEA/D-DRAM. MAMODE and IMOEA/D-CH are MO approaches. Solutions of MAMODE are dominated by solutions of MOEA/D-DRAM. Solutions of IMOEA/D-CH roughly lie on the approximation front of MOEA/D-DRAM, but the range of their solutions is much smaller than that of our solutions.

TABLE XI. SOLUTIONS OF TEST CASE 10 USING FFE BETWEEN 60000 AND 100000

Algorithm (FFE: $\times 10^4$)	Type	Objective	Best Cost	Best Emission	Best Compromise
PSOCS (6)	SO	f_{cost}			2.5269
		$f_{emission}$			2.9800
ITSA (8)	SO	f_{cost}			2.5114
		$f_{emission}$			2.9768
IBFA (8)	SO	f_{cost}	2.4817	2.6143	2.5171
		$f_{emission}$	3.2750	2.9583	2.9904
PSOAWL (10)	SO	f_{cost}			2.5269
		$f_{emission}$			2.9800
NEHS (10)	SO	f_{cost}			2.5335
		$f_{emission}$			2.9508
MOEA/D-DRAM (10)	MO	f_{cost}	**2.4712**	**2.5948**	2.5059
		$f_{emission}$	**3.2228**	**2.9282**	3.0116

TABLE XII. SOLUTIONS OF TEST CASE 10 USING 200000 OR MORE FFE

Algorithm (FFE: $\times 10^4$)	Type	Objective	Best Cost	Best Emission	Best Compromise
MAMODE (20)	MO	f_{cost}	2.4925	2.5816	2.5141
		$f_{emission}$	3.1512	2.9524	3.0274
IMOEA/D-CH (20)	MO	f_{cost}	2.4791	2.5770	2.5167
		$f_{emission}$	3.1096	2.9292	2.9780
MONNDE (n/a)	SO	f_{cost}			2.5579
		$f_{emission}$			2.9522
EFDE (40)	SO	f_{cost}			2.5327
		$f_{emission}$			2.9499
DE-SQP (120)	SO	f_{cost}	**2.4659**		2.4688
		$f_{emission}$	**3.2405**		3.1564
MOEA/D-DRAM (20)	MO	f_{cost}	2.4674	**2.5910**	2.4958
		$f_{emission}$	3.2507	**2.9221**	3.0317

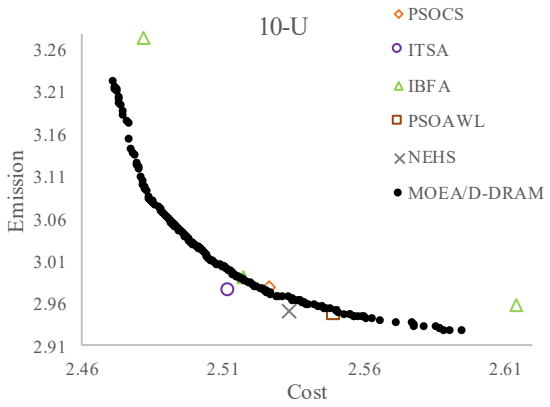


Fig. 8. Solutions of test case 10 using FFE between 60000 and 100000

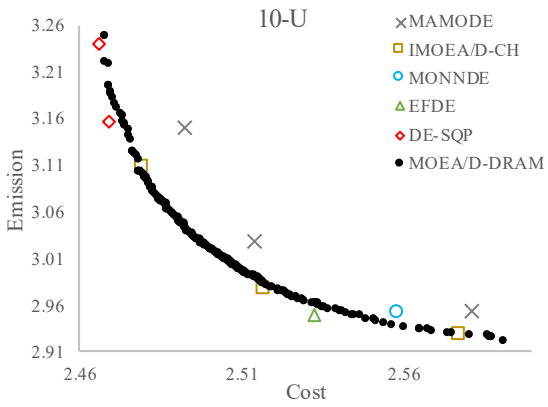


Fig. 9. Solutions of test case 10 using 200000 or more FFE

V. Conclusions and Future Work

In this paper we addressed the DEED problem through evolutionary multiobjective optimization. The DEED problem has three features: (1) it is a real-parameter optimization problem; (2) it has three types of constraints; (3) it has two objective functions. We used DE operators and proposed a dynamic mutation selection mechanism to generate solutions. We adopted a repair mechanism to make solutions satisfy all constraints with a high successful rate. The above mechanisms were integrated into the MOEA/D-DRA framework to effectively seek for the set of solutions that are non-dominated in terms of cost and emissions. Users can then observe the trade-off between the objectives and select the desired solution. Effects of the core components of our algorithm were examined through testing on six test cases and evaluating by a multiobjective indicator (IGD) and visualization. This was rarely done in the DEED literature. Experimental results confirmed positive impact of these components on algorithm performance. Finally, we compared our algorithm with 15 algorithms. When comparing with single-objective algorithms, our algorithm can find comparable solutions as well as a set of trade-off solutions in a single run; when comparing with multiobjective algorithms, our algorithm often found solutions with better convergence and distribution.

Although our algorithm works as a successful integration of several effective components, it is accompanied with many parameters. In our future work, the first task is to examine the impact of parameter setting and to provide some guidelines of parameter setting. According to the experimental results, our algorithm did not always perform well. For example, MOEA/D performed worse than NSGA-II in solving the 6-U test case, and DRA did not improve the performance in solving the 40-U test case. The second task is to do more investigation and fix these conditions. One more research direction is to apply our algorithm to solve complex DEED problems, such as those including wind energy [23] or electric vehicles [24].

Acknowldgement

This research is supported by the Ministry of Science and Technology, Taiwan, R.O.C. under Grant no. 110-2221-E-003-017.

References

[1] M. Basu, “Particle swarm optimization based goal-attainment method for dynamic economic dispatch problem,” Electric Power Components and Systems, vol. 34, pp. 1015–1025, 2006.

[2] M. Basu, “Dynamic economic emission dispatch using nondominated sorting genetic algorithm-II,” International Journal of Electrical Power & Energy Systems, vol. 30, pp. 140–149, 2008.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A feast and elitist multi-boective genetic algorithm: NSGA-II,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002. [NSGA-II]

[4] K. Deb and R. B. Agrawal, “Simulated binary crossover for continuous searchspace,” Complex Systems, vol. 9, no. 2, pp. 115–148, 1995.

[5] X. Jiang, J. Zhou, H. Wang, and Y. Zhang, “Dynamic environmental economic dispatch using multiobjective differential evolution algorithm with expanded double selection and adaptive random restart,” International Journal of Electrical Power & Energy Systems, vol. 49, pp. 399–407, 2013. [MAMODE]

[6] K. Mason, J. Duggan, and E. Howley, “Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants,” Neurocomputing, vol. 270, pp. 188–197, 2017. [PSO-AWL/PSO-GIDN]

[7] K. Mason, J. Duggan, and E. Howley, “A multi-objective neural network trained with differential evolution for dynamic economic emission dispatch,” Electrical Power and Energy Systems, vol. 100, pp. 201–221, 2018. [MONNDE]

[8] C. X. Guo, J. P. Zhan, and Q. H. Wu, “Dynamic economic emission dispatch based on group search optimizer with multiple producers,” Electric Power Systems Research, vol. 86, pp. 8–16, 2012. [GSOMP]

[9] P. K. Roy and S. Bhui, “A multi-objective hybrid evolutionary algorithm for dynamic economic emission load dispatch,” International Transactions on Electrical Energy Systems, vol. 26, pp. 49–78, 2016. [HCRO]

[10] X. Shen, D. Zou , N. Duan, and Q. Zhang, “An efficient fitness-based differential evolution algorithm and a constraint handling technique for dynamic economic emission dispatch,” Energy, vol. 186, 2019. [EFDE]

[11] S. Qian, H. Wu, and G. Xu, “An improved particle swarm optimization with clone selection principle for dynamic economic emission dispatch,” Soft Computing, vol. 24, pp. 15249–15271, 2020. [PSOCS]

[12] A. M. Elaiw, X. Xia, and A. M. Shehata, “Hybrid DE-SQP and hybrid PSO-SQP methods for solving dynamic economic emission dispatch problem with value-point effects,” Electric Power Systems Research, vol. 103, pp. 192–200, 2013. [SQP]

[13] B. Qiao, J. Liu, and X. Hao, “A multi-objective differential evolution algorithm and a constraint handling mechanism based on variables proportion for dynamic economic emission dispatch problems,” Applied Soft Computing, vol. 108, 2021. [NSDESa_LS-PDAD]

[14] H. Zhang, D. Yue, X. Xie, S. Hu, and S. Weng, “Multi-elite guide hybrid differential evolution with simulated annealing technique for dynamic economic emission dispatch,” Applied Soft Computing, vol. 34, pp. 312–323, 2015. [MOHDE-SAT]

[15] Z. Zhu, J. Wang, and M. H. Baloch, “Dynamic economic emission dispatch using modified NSGA-II,” International Transactions on Electrical Energy Systems, vol. 26, pp. 2684–2698, 2016. [MNSGA-II]

[16] 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. [MOEA/D-DE]

[17] Y. Zhu, B. Qiao, Y. Dong, B. Qu, and D. Wu, “Multiobjective dynamic economic emission dispatch using evolutionary algorithm based on decomposition,” IEEJ Transactions on Electrical and Electronic Engineering, vol. 14, pp. 1323–1333, 2019. [IMOEA/D-CH]

[18] L. Li, Z. Liu, M. Tseng, S. Zheng, and M. K. Lim, “Improved tunicate swarm algorithm: solving the dynamic economic emission dispatch problems,” Applied Soft Computing, vol. 108, 2021. [ITSA]

[19] N. Pandit, A. Tripathi, S. Tapaswi, and M. Pandit, “An improved bacterial foraging algorithm for combined static/dynamic,” Applied Soft Computing, vol. 12, pp. 3500–3513, 2012. [IBFA]

[20] Z. Li, D. Zou, and Z. Kong, “A harmony search variant and a useful constraint handling method for the dynamic economic emission dispatch problems considering transmission loss,” Engineering Applications of Artificial Intelligence, vol. 84, pp. 18–40, 2019. [NEHS]

[21] Q. Zhang, W. Liu, and H. Li, “The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances,” IEEE Congress on Evolutionary Computation, pp. 203–208, 2009. [MOEA/D-DRA]

[22] K. Deb and M. Goyal, “A combined genetic adaptive search (GeneAS) for engineering design,” Journal of Computer Science and Informatics, vol. 26, pp. 30–45, 1996.

[23] L.-L. Li, Q. Shen, M.-L. Tseng, and S. Luo, “Power system hybrid dynamic economic emission dispatch with wind energy based on improved sailfish algorithm,” Journal of Cleaner Production, vol. 316, 2021.

[24] H. Liang, Y. Liu, F. Li, and Y. Shen, “Dynamic economic/emission dispatch including PEVs for peak shaving and valley filling,” IEEE Transac

[25] tions on Industrial Electronics, vol. 66, no. 4, pp. 2880–2890, 2018.