A Multi-constraint Handling Technique based Niching Evolutionary Algorithm for Constrained Multi-objective Optimization Problems

Zixu Wang
School of Computer Science and Technology
Xidian University
Xi'an, China
cnwangzx@foxmail.com

Jingxuan Wei
School of Computer Science and Technology
Xidian University
Xi'an, China
wjx@xidian.edu.cn

Yi Zhang
School of Computer Science and Technology
Xidian University
Xi'an, China
auvyisen@gmail.com

***Abstract*—When solving constrained multi-objective optimization problems, the challenge is that how to deal with all kinds of constraints regardless of the shape of the feasible region. Especially when the feasible region is discrete or very small, some constraint handling techniques cannot solve it exactly. To address this issue, this paper proposes a new technique to handle constraints. First, all the constraints will be sorted to some grades from hard to easy according to their constrained violations. Second, a niching crowding distance mechanism is used to guarantee the diversity of the pareto front better. The experiments show that the proposed algorithm can generate a set uniformly distributed pareto optimal solutions under constrains.**

Keywords—constraints, evolutionary algorithm, multi-objective, optimization

I. INTRODUCTION

The constrained optimization problems (COPs) exist widely in the real world. COPs usually contains multiple conflicting objectives and a series of constraints. According to the number of objectives, it can be divided into constrained multi-objective optimization problems (CMOPs) [1] and constrained many-objective optimization problems [2]. The number of objectives of the former is less than or equal to three, while the latter is more than three. Generally speaking, a CMOP can be formulated as:

$$\begin{aligned} \text{minimize} \quad & F(\mathrm{x}) = \left(f_1(\mathrm{x}), \ldots f_2(\mathrm{x}), f_m(\mathrm{x})\right)^T \\ \text{subject to} \quad & g_j(\mathrm{x}) \geq 0, j = 1, \ldots, p \\ & h_k(\mathrm{x}) = 0, k = 1, \ldots, q \\ & \mathrm{x} \in \Omega \end{aligned}$$

Where $\mathrm{x}=\left(x_1, x_2, \ldots, x_n\right)^{\mathrm{T}}$ is an n-dimensional decision vector. $\Omega=\left[x_i^L, x_i^U\right]^n \subseteq \mathbb{R}$ represents the decision space. $F: \Omega \rightarrow \mathbb{R}$ including m conflicting objective functions where $\mathbb{R}$ is the objective space. $g_j(\mathrm{x})$ is the j-th inequality constraint while $h_k(\mathrm{x})$ is the k-th equality constraint. Next, some definitions about CMOP are introduced briefly.

The constraint violation (CV): first, the degree of violation of x in the j-th inequality constraint and k-th equality constraint can be defined as $\mathrm{G}_j(\mathrm{x}) = \left|\min\left\{0, g_j(\mathrm{x})\right\}\right|, j = 1, \ldots, p$ and $H_k(\mathrm{x}) = \max\left\{0, \left|h_k(\mathrm{x})\right|\right\}, k = 1, \ldots, q$ respectively. Thus, the CV is defined as:

$$CV(\mathrm{x}) = \sum_{j=1}^{p} G_j(\mathrm{x}) + \sum_{k=1}^{q} H_k(\mathrm{x})$$

Feasible solution: x is a feasible solution if and only if it satisfies all constraints which means $CV(\mathrm{x}) = 0$.

Infeasible solution: x is an infeasible solution as long as it does not satisfy any constraints which means $CV(\mathrm{x}) > 0$.

Pareto dominance: x_1 and x_2 are two solutions that belong to Ω. If $f_i(\mathrm{x}_1) \leq f_i(\mathrm{x}_2)\ \forall i \in \{1, 2..., m\}$ and $\exists i \in \{1, 2..., m\}$ that make $f_i(\mathrm{x}_1) < f_i(\mathrm{x}_2)$, we call x_1 dominates x_2, denoted as $\mathrm{x}_1 \prec$. Moreover, if there is no solution dominating x_1 (x_1 is a feasible solution), then x_1 is called a pareto optimal.

Pareto set (PS): Pareto set contains all pareto optimal solutions.

Pareto front (PF): PF is the image of all pareto optimal solutions.

Next, we will give a brief review of unconstrained multi-objective evolutionary algorithms.

MOEAs can be classified into three categories: dominance-based, decomposition-based and indicator-based. One common feature of these algorithms is that they all focus on the convergence and diversity of population. 1) dominance-based EAs ranking individuals by non-dominant relations such as NSGAII [3] which use nondominated sorting to represent convergence and then use crowing distance assignment to represent diversity. 2)decomposition-based EAs such as MOEA/D [4] decompose a MOP into a series of single-objective subproblems according to uniformly distributed weight vectors in space. The convergence can be guaranteed by optimizing each single objective, and the diversity can be guaranteed by uniformly distributed vectors. There are many variants of MOEA/D such as MOEA/DD [5], MOEA/D-M2M [6] and MOEA/D-AWA [7]. 3)indicator-based EAs use performance indicators to evaluate individuals of each generation. The performance of the algorithms will be affected by the different emphasis of the indicator. For example, IBEA [8] used $I_{\varepsilon+}$ as indicator

This work was supported by the National Science Foundation of China under Grant 61203372. *(Corresponding author: Jingxuan Wei.)*

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

which is a convergence indicator, so IBEA performs poorly in diversity [9]. Hypervolume indicator concern both convergence and diversity but it has high computational complexity. As a result, many indicator-based EAs that use hypervolume as their indicator such as HypE [10] are computationally heavy and very slow.

However, compared with the prosperity of unconstrained multi-objective evolutionary algorithms, the research of constrained multi-objective evolutionary algorithms is few. Most of the existing algorithms use the combination of a well performed unconstrained algorithm and a constraint handling technique. In this case, the performance of the constraint handling technique will affect the performance of the algorithm. We will give details of existing constraint handling techniques in Section II.

In order to address the limitations, the goal of this paper is to develop an evolutionary algorithm to find the pareto front under constraints. The pareto optimal solutions obtained under constraints are expected to have good performance in terms of convergence and diversity. To achieve the goal, we propose two new techniques:

1) A new constraint handling technique that deal with multi-constraints is proposed, denoted as MC-CHT. This technique can enlarge the feasible region during the early stage of evolution and then gradually reduce to true size. Moreover, we hybrid this method with ε-constraint method.
2) A new niching method is used to maintain the diversity of population. This method can avoid two extreme situations in original crowding distance [3].

The rest of this paper is organized as follows. Section II overviews existing constrained handling techniques. Then explain the motivations of new constraint handling technique. Section III shows the details of the proposed constrained handling technique and the whole algorithm. Section IV introduces the general experimental setting and shows specific experiment results that compared with four popular CMOEAs and analyzes the experiment results. Section V gives the conclusion and future direction.

II. PRELIMINARIES

A. Existing constraint handling techniques

Constraint handling is a crucial part of CMOPs. All these constraint handling techniques can be divided into three categories.

1) The first category always give priority to feasible solutions when choosing promising solutions into next generation. The typical methods include penalty function methods [11], constrained-domination principle [12], and ε-constraint method [13]. The main idea of penalty function is adding a penalty term that connect with CV to original objectives. A solution with high CV will have high penalty than others with low CV, of course, feasible solutions have no penalty. Constrained -domination principle transforms normal dominance relation that defined before to constrained dominance relation. Specifically, there are two solutions x_1 and x_2, we call x_1 constrained-dominate x_2 if any conditions are satisfied: 1) x_1 is feasible solution while x_2 is infeasible solution; 2) x_1 and x_2 are both feasible solutions but $x_1 \prec$; 3) x_1 and x_2 are both infeasible solutions but $CV(x_1) < CV(x_2)$. ε -constraint method is very similar to CDP except for one parameter ε . Specifically, if the CV of an infeasible solution is less than ε , then we think of it as a feasible solution. Otherwise, it is still an infeasible solution. After redefining the feasibility of the solutions, CDP is then used to determine the constrained-dominance relation.
2) The second category tries to balance the trade-off between feasibility and convergence. In [14], stochastic ranking is proposed. When comparing two solutions, stochastic ranking will decide the winner randomly according to its value of objectives or constraint violation. [15] proposes a multi-objective-based method that transform constraint into objective and so a CMOP is transformed into a MOP. Recently, a Tri-Goal framework [16] based on BiGE [17] is proposed to balance convergence, diversity and feasibility. In this framework, convergence, diversity and feasibility of a solution are regarded as three objectives for non-dominated sorting.
3) The third category is to hybrid several constraint handling techniques properly in order to get a better performance. [18] proposes an ensemble of constraint handling techniques to solve COPs, where each constraint handling technique has its own population and can learn from other populations to deal with complicate problems. [19] proposes (μ + λ)-constrained differential evolution that hybrids differential evolution with three mutation strategies and improved adaptive trade-off model which uses three CHTs to handle three types of situation. [20] hybrids CDP and ε-constrained method in a DE framework.

B. Motivation

We can intuitively conclude that the more constraints, the more difficult it is to deal with. Under this consideration, we can deal with constraints one by one. Before formal explanation, we propose a new definition called constraint capability:

Constraint capability: for example, there are two constraints C_a and C_b. Each constraint determines part of the infeasible region. Let us say R_a is the infeasible region determined by C_a and R_b is the infeasible region determined by C_b. If the area or hypervolume of R_a is larger than R_b, then we say that C_a has stronger constraint capability than C_b.

When the constraints are many and complex, the real feasible region may be very narrow. It is hard for the population to find the right direction of evolution. However, when the algorithm only deals with one constraint, the feasible region will be much larger. After dealing with one constraint, then we add new constraints one by one, in doing so, the population can gradually approximate the real pareto front in the real feasible region. Considering the convergence rate, our proposed technique (MC-CHT) will deal with constraints from difficult to easy according to the constraint capability.

III. PROPOSED ALGORITHM

The algorithm proposed in this paper is called MC/MOEA. The details of the whole algorithm are described in the following.

A. General framework

The main procedure is described in Algorithm 1. First, constrained sorting is used to get the sorted constraints according to constraint capability. Then, an initial population P_0 is generated randomly. Before the main loop, a procedure named environmentSelection is used to get f_c, f_d and f_f of each solution. f_c, f_d and f_f are values that can reflect the convergence, diversity and feasibility of solutions in population. In the main loop, we get P' by select promising parents according to their f_c, f_d and f_f. Then, after variation operators, we get P''. Finally, we combine P' and P'', and choose next generation by environmentSelection.

Algorithm 1: general framework

Input: population size N, parameter N_{MC}, maximum iteration

Output: population P

1. sorted constraints ← constrained sorting (N_{MC}) (see III-B, this procedure sort constraints according to constraint capability)
2. P_0 ← initialization (N)
3. t ← 0
4. [~, f_c, f_d, f_f] ← environmentSelection (P_t, sorted constraints, N) (see III-C)
5. while not terminated do
6. P' ← matingSelection (f_f, f_c, f_d) (see III-F)
7. P'' ← variation (P')
8. S ← $P' \bigcup$
9. [P_{t+1}, f_c, f_d, f_f] ← environmentSelction (S, sorted constraints, N)
10. t ← t+1
11. end

B. Constrained sorting

This procedure only needs one parameter, which is the number of generated solutions, denoted as N_{MC}. In this paper, we set it as 1000 by experiments. After generating these solutions, we calculate feasible proportion (fp) and constraint violation (CV) of all solutions under each constraint. Finally, we sort the constraints by feasible proportion and CV. The smaller the feasible proportion, the greater the constraint capability. If two constraints have same feasible proportion, then we will compare constraint violation of them. The rules are like feasible proportion. The details are showed in Algorithm 2.

Noted that constraint violation CV_i means constraint violation under constraint C_i, where C_i is one constraint.

Algorithm 2: constrained sorting

Input: parameter N_{MC}, the number of constraints N_c

Output: sorted constraints (SC)

1. P_{MC} ← initialization (N_{MC}) (randomly generate solutions)
2. feasibleProportion (fp_i) ← N_{MC}, CV_i ← 0 for i = 1, …, N_c
3. for each constraint i (C_i) do
4. for each solution x do
5. if $C_i(x) > 0$ then
6. fP_i -= 1
7. CV_i += $C_i(x)$ ($C_i(x)$ means constraint violation of x under constraint i)
8. end
9. end
10. end
11. sorted constraints ← sort (fp_i, CV_i)

C. EnvironmentSelection

This procedure can calculate f_c, f_d and f_f of solutions and then choose promising solutions into next generation according to the values. The details are showed in Algorithm III. First, MC-CHT is used to calculate the current constraint violation (denoted as f_f) of each solution. Second, non-dominated sorting with constrained-dominance principle is used to calculate level (denoted as f_c) of each solution. F_l includes solutions that have same level l. Third, a niching method is used to calculate f_d. Finally, we choose N solutions according to f_c and f_d.

D. MC-CHT

This operation is to enlarge the feasible region in the stage of evolution. And ε-constraint can also achieve this goal. So, this paper hybrid ε-constraint method to enhance the ability of MC-CHT. First, the number of constraints that need to be used in the current generation is calculated as follows:

$$\mathrm{d} = \left\lceil t/(\varphi * T_{\max}) \right\rceil * N_c$$

where t is the current generation, $T_{\max}$ is the maximum of generation and N_c is the total number of constraints. φ is the maximum of generation that use MC-CHT. Second, the constraint violation of solutions will be calculated as follows:

$$CV(x) \leftarrow \sum_{i=1}^{d} CV_i(x)$$

where $CV_i(x)$ is the constraint violation of x under ith constraint in the sorted constraints. Third, ε-constraint will be used to calculate f_f of solutions. If solution x are

feasible under ε-constraint, then the f_f equals to zero, otherwise, the f_f equals to $CV(x)$.

Algorithm 3: environmentSelection

Input: population P_t, population size N, φ, sorted constraints (SC)

Output: population Q, f_c, f_d, f_f

```
Q ← ∅, l ← 1
f_f ← MC-CHT (P_t, φ, SC, N) (see III-D)
f_c ← constrained non-dominated sorting (P_t, f_f)
f_d ← niching (P_t) (see III-E)
while |Q| ≤ N do
    Q ← Q ∪ F_l
    l ++
end
if |Q| > N then
    Q ← Q - F_l
end
P_next = sort (f_d (F_l))
Q ← P_next
```

E. Niching method

In NSGAII, when two solutions are close together in the same level, two extreme situations may occur. Fig 1 and Fig 2 show the two extreme situations. In Fig.1, solution E and F have bigger crowding distance than solution B and C (f_1 and f_2 are objectives). In this case, neither B or C can be chosen to be parents or into next generation. In Fig.2, when solution D and G are closer with E and F, situation are different. In this case, neither E or F can be chosen to be parents or into next generation. In either case, it will lead to a serious loss of population diversity. Under this consideration, we use a niching method to improve original crowding distance. In this paper, we use $1/\sqrt[M]{N}$ to determine whether the two solutions are close, where N is the number of solutions and M is the number of objectives. When some solutions are too close, we set f_d of solutions to zero except only one. The details are showed in Algorithm 4.

F. MatingSelection

The matingSelection is used to choose promising parents from population. When two solutions are compared, solution with better feasibility will be selected. If they have same feasibility, then we will choose solution that have better convergence.

IV. EXPERIMENT RESULTS

A. Experimental setup

1) Test problems: DAS-CMOPs [21] are chosen as the

Algorithm 4: niching method

Input: population P_t, $r\,(1/\sqrt[M]{N})$

Output: diversity f_d

```
R ← Calculate the Euclidean distance between two solutions R_ij is distance between i and j.
for i = 1: N do
    Nei_i ← i
end
for i = 1: N do
    for j = i +1: N do
        if R_ij ≤ r then
            Nei_i ← j
            Nei_j ← i
        end
    end
end
for i = 1: N do
    if f_d (i) == 0
        continue
    for each neighbor j in Nei_i do
        if Nei_i ≠ Nei_j then
            Break
    end for
    for all neighbor j do
        f_d (j) ← 0
end
```

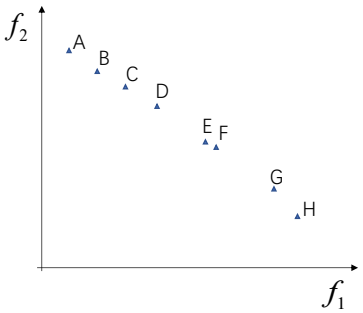

Fig.1. Situation that solution E and F both have high priority

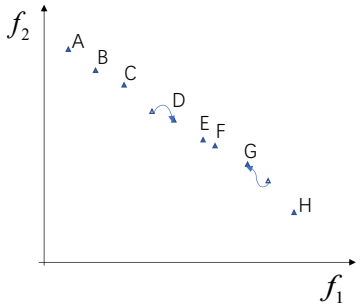

Fig.2 Situation that solution E and F both have low priority

TABLE I. STATISTICAL RESULTS (MEAN AND STANDARD DEVIATION) OF THE HV VALUES OBTAINED BY PEER ALGORITHMS FOR DASCMOP PROBLEMS.

Problem	CMOEAD	ANSGAIII	KnEA	ARMOEA	MC/MOEA
DASCMOP1	4.0592e-3 (2.61e-3) =	5.6373e-3 (5.82e-3) =	5.7262e-3 (5.44e-3) =	2.9035e-3 (3.81e-3) -	9.0480e-3 (1.56e-2)
DASCMOP2	2.4340e-1 (1.03e-2) +	2.3568e-1 (9.73e-3) =	2.2618e-1 (6.89e-3) -	2.3202e-1 (8.33e-3) -	2.3571e-1 (9.24e-3)
DASCMOP3	1.9679e-1 (4.97e-2) -	2.0837e-1 (1.10e-2) -	1.9516e-1 (4.62e-2) -	1.9564e-1 (4.60e-2) -	2.2054e-1 (1.58e-2)
DASCMOP4	2.0370e-2 (2.62e-2) -	2.4308e-2 (1.50e-2) -	3.0582e-2 (1.37e-2) -	2.4228e-2 (1.20e-2) -	5.1233e-2 (3.23e-2)
DASCMOP5	6.8806e-2 (7.33e-2) -	5.1422e-2 (5.31e-2) -	5.4822e-2 (4.73e-2) -	5.7267e-2 (7.67e-2) -	1.4700e-1 (1.13e-1)
DASCMOP6	4.2500e-2 (6.06e-2) -	2.5569e-2 (2.65e-2) -	1.9456e-2 (1.49e-2) -	1.9106e-2 (1.86e-2) -	1.0370e-1 (9.46e-2)
DASCMOP7	1.7122e-1 (5.96e-2) -	2.1748e-1 (2.65e-2) =	2.3005e-1 (2.79e-2) =	2.3391e-1 (3.33e-2) =	2.2609e-1 (3.86e-2)
DASCMOP8	8.8443e-2 (4.74e-2) -	1.3728e-1 (2.92e-2) =	1.5756e-1 (2.47e-2) +	1.4372e-1 (3.60e-2) =	1.2995e-1 (3.26e-2)
DASCMOP9	9.6557e-2 (2.55e-2) =	9.7584e-2 (1.31e-2) -	9.3521e-2 (1.18e-2) -	9.5820e-2 (1.37e-2) -	1.0587e-1 (1.66e-2)
+/-/=	1/6/2	0/5/4	1/6/2	0/7/2	

[a.] "+", "-", "=" indicate that the results are significantly better, worse or similar to that obtained by MC/MOEA under rank sum test, respectively.

TABLE II. STATISTICAL RESULTS (MEAN AND STANDARD DEVIATION) OF THE IGD VALUES OBTAINED BY PEER ALGORITHMS FOR DASCMOP PROBLEMS.

Problem	CMOEAD	ANSGAIII	KnEA	ARMOEA	MC/MOEA
DASCMOP1	7.2504e-1 (1.46e-2) =	7.2283e-1 (2.99e-2) =	7.2493e-1 (2.98e-2) =	7.3820e-1 (2.26e-2) -	7.1328e-1 (4.01e-2)
DASCMOP2	2.7617e-1 (3.78e-2) +	2.9307e-1 (2.73e-2) =	3.3381e-1 (2.72e-2) -	3.0829e-1 (2.77e-2) =	2.9928e-1 (2.73e-2)
DASCMOP3	3.7632e-1 (1.07e-1) -	3.4672e-1 (3.20e-2) -	3.8748e-1 (9.14e-2) -	3.7997e-1 (9.12e-2) -	2.9251e-1 (5.11e-2)
DASCMOP4	6.7654e-1 (1.87e-1) -	6.0047e-1 (1.03e-1) -	5.5467e-1 (9.69e-2) -	5.9609e-1 (7.22e-2) -	4.7648e-1 (9.38e-2)
DASCMOP5	6.1370e-1 (2.24e-1) -	6.7304e-1 (1.81e-1) -	6.1807e-1 (1.34e-1) =	6.5090e-1 (1.90e-1) -	4.1289e-1 (2.76e-1)
DASCMOP6	6.9869e-1 (2.06e-1) -	7.1496e-1 (1.32e-1) -	7.6614e-1 (1.83e-1) -	7.5042e-1 (1.13e-1) -	5.0755e-1 (2.02e-1)
DASCMOP7	2.7333e-1 (1.63e-1) -	1.6858e-1 (6.64e-2) =	1.5142e-1 (8.06e-2) =	1.4247e-1 (8.72e-2) =	1.4933e-1 (9.12e-2)
DASCMOP8	4.3384e-1 (2.58e-1) -	2.6775e-1 (1.47e-1) =	1.8022e-1 (9.28e-2) +	2.6501e-1 (1.74e-1) =	2.4411e-1 (1.10e-1)
DASCMOP9	4.7719e-1 (1.48e-1) =	4.7589e-1 (6.31e-2) =	4.8661e-1 (6.23e-2) -	4.9047e-1 (7.67e-2) =	4.5258e-1 (8.60e-2)
+/-/=	1/6/2	0/4/5	1/5/3	0/5/4	

[b.] "+", "-", "=" indicate that the results are significantly better, worse or similar to that obtained by MC/MOEA under rank sum test, respectively.

test problems for its difficulty-adjustable and scalable. In this paper, we use (0.5,0.5,0.5) as the parameter.

2) Indicators: Two popular indicators IGD [22] and HV [23] are chosen to compare the performance of algorithms. Both of them can evaluate convergence and diversity simultaneously.

*3) Comparison algorithms:*In this paper, we choose four popular algorithms: KnEA-CDP [24], CMOEA/D [4], ARMOEA-CDP [25], and ANSGAIII [26].

4) General experimental setting: Each algorithm runs independently 30 times on each problem and maximum function evaluations of each run are 30,000. The results will be tested by Wilcoxon rank sum test at 5% significance level. All algorithms use genetic operator (GA) with simulated binary crossover (SBX) and polynomial mutation (PM) to produce offspring. The probability and distribution of SBX are set to 1 and 20, while the probablity of PM is set to 1/n (n is the dimension of decision vector). φ is set to 0.8 $T_{\max}$. For MOEA/D, the size of neighborhood is set to 10 and PBI approach is used.

B. Experiment results

Table 1 and table 2 show the results of five peer algorithms on HV and IGD, where the best results are highlighted. From the table, we can find that MC/MOEA performs best on DASCMOP1, DASCMOP3-6 and DASCMOP9 whether on IGD or HV. Especially on DASCMOP5-6, MC/MOEA performs particularly well on HV indicator. CMOEA/D performs best on DASCMOP2 whether on IGD and HV. It should be noted that the performance of MC/MOEA is a little poor on this problem compared with CMOEA/D. KnEA performs best on DASCMOP8 whether on IGD or HV, while MC/MOEA performs second worst on HV. But MC/MOEA performs second best on IGD. ARMOEA performs best on DASCMOP7 whether on IGD or HV, while MC/MOEA performs second best on IGD. Overall, MC/MOEA performs significantly better than other algorithms on DASCMOP1-9.

V. CONCLUSION

The goal of this paper was to propose an evolutionary algorithm that can generate a set of well distributed optimal solutions under constraints. And the goal was successfully achieved by two new techniques: one is a new constraint handling technique that can enlarge the feasible region in the early stage of evolution. The other is a new niching method that can maintain diversity better. The results showed that the proposed algorithm has high competitiveness compared with four popular algorithms. In the future, we still have a lot to expand in our work. For example, we can improve MC-CHT to accommodate different number of constraints. Moreover, we can apply MC-CHT to constrained many-objective optimization problems.

REFERENCES

[1] C. A. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," in *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28-36, Feb. 2006.

[2] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

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

[4] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," in *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, Dec. 2007.

[5] K. Li, K. Deb, Q. Zhang and S. Kwong, "An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition," in *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694-716, Oct. 2015.

[6] H. Liu, F. Gu and Q. Zhang, "Decomposition of a Multiobjective Optimization Problem Into a Number of Simple Multiobjective Subproblems," in *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450-455, June 2014.

[7] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun and J. Wu, "MOEA/D with Adaptive Weight Adjustment," in *Evolutionary Computation*, vol. 22, no. 2, pp. 231-264, June 2014.

[8] E. Zitzler and S. K˙unzli, "Indicator-based selection in multiobjective search," in PPSN'04: Proc. of 8th International Conference on Parallel Problem Solving from Nature - PPSN VIII, 2004, pp. 832–842

[9] D. Hadka and P. Reed, "Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization," in *Evolutionary Computation*, vol. 20, no. 3, pp. 423-452, Sept. 2012.

[10] J. Bader and E. Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," in *Evolutionary Computation*, vol. 19, no. 1, pp. 45-76, March 2011.

[11] C. Saha, S. Das, K. Pal and S. Mukherjee, "A Fuzzy Rule-Based Penalty Function Approach for Constrained Evolutionary Optimization," in *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2953-2965, Dec. 2016.

[12] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[13] Z. Fan *et al*., "An improved epsilon constraint handling method embedded in MOEA/D for constrained multi-objective optimization problems," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, 2016, pp. 1-8.

[14] T. P. Runarsson and Xin Yao, "Stochastic ranking for constrained evolutionary optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284-294, Sept. 2000.

[15] Z. Cai and Y. Wang, "A Multiobjective Optimization-Based Evolutionary Algorithm for Constrained Optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658-675, Dec. 2006.

[16] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang and J. Zhang, "Tri-Goal Evolution Framework for Constrained Many-Objective Optimization," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

[17] M. Li, S.Yang, and X. Liu, "Bi-goal evolution for manyobjective optimizaton problems," Artif. Intell., vol. 228, pp. 45–65, Nov. 2015.

[18] R. Mallipeddi and P. N. Suganthan, "Ensemble of Constraint Handling Techniques," in *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 561-579, Aug. 2010.

[19] Y. Wang and Z. Cai, "Constrained Evolutionary Optimization by Means of $(\mu + \lambda)$-Differential Evolution and Improved Adaptive Trade-Off Model," in *Evolutionary Computation*, vol. 19, no. 2, pp. 249-285, June 2011.

[20] S.M. Elsayed, R. A. Sarker and D. L. Essam, "Integrated strategies differetial evolution algorithm with a local search for constrained optimization," *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, 2011, pp. 2618-2625.

[21] Z Fan et al., "Difficulty adjustable and scalable constrained multiobjective test problem toolkit," arXiv preprint arXiv:1612.07603, 2016.

[22] Y Sun, G. G. Yen and Z. Yi, "IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems," in *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 173-187, April 2019.

[23] L. M. S. Russo and A. P. Francisco, "Quick Hypervolume," in *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 481-502, Aug. 2014.

[24] X. Zhang, Y. Tian and Y. Jin, "A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 761-776, Dec. 2015.

[25] Y. Tian, R. Cheng, X. Zhang, F. Cheng and Y. Jin, "An Indicator-Based Multiobjective Evolutionary Algorithm With Reference Point Adaptation for Better Versatility," in *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609-622, Aug. 2018.

[26] H. Jain and K. Deb, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach," in *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602-622, Aug. 2014.