# Epsilon Constrained Method for Constrained Multiobjective Optimization Problems: Some Preliminary Results

Zhixiang Yang, Xinye Cai,Zhun Fan
The College of Computer Science and Technology, The Department of Electrical Engineering
Nanjing University of Aeronautics and Astronautics, Shantou University
xiang052@163.com, xinye@nuaa.edu.cn, zfan@stu.edu.cn

## ABSTRACT

In this paper, the $\boldsymbol{\varepsilon}$ constrained method and Adaptive operator selection (AOS) are used in Multiobjective evolutionary algorithm based on decomposition (MOEA/D). The $\boldsymbol{\varepsilon}$ constrained method is an algorithm transformation method, which can convert algorithms for unconstrained problems to algorithms for constrained problems using the $\boldsymbol{\varepsilon}$ level comparison, which compares search points based on the pair of objective value and constraint violation of them. AOS is used to determine the application rates of different operators in an online manner based on their recent performances within an optimization process. The experimental results show our proposed approach for multiobjective constrained optimization is very competitive compared with other state-of-art algorithms.

## Keywords

Multiobjective evolutionary algorithm based on decomposition (MOEA/D), Adaptive operator selection, $\varepsilon$ Constrained method, Constrained optimization problems

## 1. INTRODUCTION

This paper considers the following constrained multiobjective optimization problem:

$$\begin{aligned} \text{minimize} \quad & F(x) = (f_1(x), \ldots, f_m(x))^T \\ \text{subject to} \quad & g_i(x) \geq 0, i = 1, \ldots, p \\ & l_i \leq x_i \leq u_i, i = 1. \ldots, n \end{aligned} \tag{1}$$

where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is an $n$ dimensional vector of decision variables, the inequalities $g_i(x) \geq 0, i = 1, \ldots, p$, are constraints. $l_i$ and $u_i$ are the lower and upper bounds of $x_i$ for $i = 1, \ldots, n$. The objective vector function $F$ consists of $m$ real-valued objective functions.

Multiobjective evolutionary algorithm based on decomposition (MOEA/D) [13] is a recent evolutionary algorithmic framework for multiobjective optimization, which explicitly decomposes a multiobjective optimization problem(MOP) into scalar optimization subproblems and solves these subproblems simultaneously by evolving a population of solutions. Its two unconstrained versions: MOEA/D-DE [11] and MOEA/D-DRA [14], work very well on unconstrained multiobjective optimization problems (e.g. [15]).

There exist many studies on solving constrained single objective optimization problems using evolutionary algorithms (EAs) [2] [8]. Among them, the $\varepsilon$ constrained differential evolution ($\varepsilon$DE) is one of the most popular algorithms. The $\varepsilon$DE shows its great performance in solving constrained single objective optimization problems. The $\varepsilon$ constrained method [5] is an algorithm transformation method, which can convert constrained optimization problems to unconstrained ones using the $\varepsilon$ level comparison, which compares search points based on the pair of objective value and constraint violation of them. In this paper, we introduce $\varepsilon$ constrained method into MOEA/D in order to extend it for constrained optimization problems.

Furthermore, in order to improve the efficiency of our approach, an adaptive operator selection (AOS) method is also adopted in our proposed approach. More specifically, a bandit-based AOS method (FRRMAB) is employed, which uses a sliding window to follow the dynamics of the search process.

The remainder of this paper is organized as follows. Section 2 introduces the related work of constrained multiobjective optimization. The following Section 3 is mainly dedicated to the detailed descriptions of our proposed algorithm. Experimental setting is given in Section 4. Experimental studies and discussions are detailed in Section 5. The final conclusions of this paper is in Section 5.

## 2. THE RELATED WORK

### 2.1 Tchebycheff Aggregation Function

MOEA/D needs to decompose a multiobjective problem into a number of single objective subproblems. In this paper, we use the Tchebycheff aggregation function for this purpose. It is defined as follows:

$$\begin{aligned} \text{minimize} \quad & g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i^*|\} \\ \text{subject to} \quad & x \in S \end{aligned} \tag{2}$$

where $S$ is the feasible region. $z^* = (z_1^*, \ldots, z_m^*)^T$ is the reference point, i.e.,$z_i^* = \min\{f_i(x) | x \in S\}$ for each $i = 1, \ldots, m$ and $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ is a weight vector, i.e., $\lambda \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^m \lambda_i = 1$.


*GECCO'14,* July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2610012 .

## 2.2 Constraint violation and $\varepsilon$ level comparisons

In the $\varepsilon$ constrained method, constraint violation $\phi(x)$ is defined. The constraint violation can be given by the maximum of all constraints or the sum of all constraints.

$$\phi(x) = \max\{\max_j\{0, g_j(x)\}, \max_j |h_j(x)|\} \quad (3)$$

$$\phi(x) = \sum_j ||\max\{0, g_j(x)\}||^p + \sum_j ||h_j(x)||^p \quad (4)$$

where $p$ is a positive number.

The $\varepsilon$ level comparisons are defined as an order relation on a pair of objective function value and constraint violation $(f(x), \phi(x))$. If the constraint violation of a point is greater than 0, the point is not feasible and its worth is low. The $\varepsilon$ level comparisons are defined basically as a lexicographic order in which $\phi(x)$ precedes $f(x)$, because the feasibility of $x$ is more important than the minimization of $f(x)$. This precedence can be adjusted by the parameter $\varepsilon$.

Let $f_1(f_2)$ and $\phi_1(\phi_2)$ be the function values and the constraint violation at a point $x_1(x_2)$, respectively. Then, for any $\varepsilon$ satisfying $\varepsilon \geq 0$, $\varepsilon$ level comparisons $<_\varepsilon$ and $\leq_\varepsilon$ between $(f_1, \phi_1)$ and $(f_2, \phi_2)$ are defined as follows:

$$(f_1, \phi_1) <_\varepsilon (f_2, \phi_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{if } \phi_1, \phi_2 \leq \varepsilon \\ f_1 < f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \quad (5)$$

$$(f_1, \phi_1) \leq_\varepsilon (f_2, \phi_2) \Leftrightarrow \begin{cases} f_1 \leq f_2, & \text{if } \phi_1, \phi_2 \leq \varepsilon \\ f_1 \leq f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \quad (6)$$

In case of $\varepsilon = \infty$, the $\varepsilon$ level comparisons $<_\infty$ and $\leq_\infty$ are equivalent to the ordinary comparisons $<$ and $\leq$ between function values. Also, in case of $\varepsilon = 0$, $<_0$ and $\leq_0$ are equivalent to the lexicographic orders in which the constraint violation $\phi(x)$ precedes the function value $f(x)$.

## 2.3 Fitness-Rate-Rank-Based Multiarmed Bandit Adaptive Operator Selection

In this section, we adopt a new bandit-based method for the to adaptively select operators. The proposed method pays particular attention to the AOS dynamic nature [9]. This consists of two modules. One is for credit assignment and the other is for operator selection.

### 2.3.1 Credit Assignment

In credit assignment, one needs, to address the following two issues:

1) how to measure the impact in the search process caused by the application of an operator;
2) how to assign an appropriate credit value to an operator based on this measured impact.

As for the first issue, the most commonly used approach is to directly use the raw values of the fitness improvements caused by the recent uses of the operator under assessment. However, the range of raw fitness improvements varies from problem to problem and even at the different stages of an optimization process. It is common that the raw fitness improvement value is much larger at early stages than at later ones. Therefore, as discussed in [3], the direct use of raw fitness improvement could deteriorate the algorithm's robustness. To alleviate this problem, our proposed method uses the fitness improvement rates (FIR). More specifically, the FIR achieved by an operator $i$ at time point $t$ is defined as

$$FIR_{i,t} = \frac{pf_{i,t} - cf_{i,t}}{pf_{i,t}} \quad (7)$$

where $pf_{i,t}$ is the fitness value of the parent, and $cf_{i,t}$ is the fitness value of the offspring. A sliding window with fixed size $W$ is used to store the FIR values of the recently used operators. It is organized as a first-in, first-out (FIFO) queue, i.e., the FIR value of the most recently used operator is added at the tail of the sliding window, while the oldest record (the item at the head of the queue) is removed to keep the window size constant. Each slot in the sliding window stores two components:

1) the index of the operator *op* used;
2) its FIR value.

The major reason for using the sliding window is that, in dynamic AOS environments, the performance of an operator in a very early stage may be irrelevant to its current performance. The sliding window ensures that the stored FIR information is for the current situation of the search.

To address the second issue set at the outset of this subsection, we first compute $Reward_i$, the sum of all FIR values for each operator $i$ in the current sliding window. Then, we rank all these $Reward_i$ values in a descending order. Let $Rank_i$ be the rank value of operator $i$, inspired by other recently proposed rank-based credit assignment schemes, and to give more chances to the best operators, we introduce a decaying factor $D \in [0, 1]$ to transform $Reward_i$ to

$$Decay_i = D^{Rank_i} \times Reward_i. \quad (8)$$

Then, we assign the following credit value to operator $i$:

$$FRR_{i,t} = \frac{Decay_i}{\sum_{j=1}^{K} Decay_j} \quad (9)$$

Clearly, the smaller the value of $D$, the larger the influence for the best operator.

Finally, at each time point t, the operator maximizing the following function is selected:

$$FRR_{i,t} + C \times \sqrt{\frac{2 \times \ln \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}} \quad (10)$$

where $C$ is a scaling factor to control the tradeoff between exploitation (the first term that favors the operators with best empirical rewards) and exploration (the square root term that favors the infrequently tried operators). $n_{i,t}$ is the number of times operator $i$ has been applied in the recent $K$ applications.

### 2.3.2 Operator Selection

Based on the received credit values, the operator selection scheme selects suitable operators to generate new solutions. This paper uses a bandit-based operator selection scheme. Our scheme is similar to that used in [3]. The major difference is that we use $FRR$ values as the quality index instead of the average of all the rewards received so far for an operator. In addition, $n_i$ indicates the number of times operator $i$ has been selected in the recent $W$ applications.

The pseudocode of our proposed bandit-based operator selection scheme is given in Algorithm 1. The combination

of this operator selection with the credit assignment schemes constitutes is named as FRRMAB in our paper. It is worth noting that no operator has yet been applied at the beginning of the search; thus, we give each operator an equal chance to be selected in this case. FRRMAB is not employed until each operator has been applied at least once.

**Algorithm 1:** The procedure for the bandit-based operator selection

1: **if** There are operators that have not been selected **then**
2: $op_t$ = one that uniformly randomly selected from the operators pool;
3: **else**
4: $op_t = \arg\max_{i=\{1,\ldots,K\}} (FRR_{i,t} + C \times \sqrt{\frac{2\times \ln \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}})$
5: **end**

# 3. THE PROPOSED ALGORITHM

## 3.1 Using FRRMAB to Enhance MOEA/D

**Operators Pool:** Many different DE mutation operators have been proposed. Four of them, which present distinct search characteristics, are chosen for the AOS in our experiments:

1) DE/rand/1

$$v^i = x^i + F \times (x^{r_1} - x^{r_2}); \quad (11)$$

2) DE/rand/2

$$v^i = x^i + F \times (x^{r_1} - x^{r_2}) + F \times (x^{r_3} - x^{r_4}); \quad (12)$$

3) DE/current-to-rand/1

$$v^i = x^i + K \times (x^i - x^{r_1}) + F \times (x^{r_2} - x^{r_3}); \quad (13)$$

4) DE/current-to-rand/2

$$v^i = x^i + K \times (x^i - x^{r_1}) + F \times (x^{r_2} - x^{r_3}) + F \times (x^{r_4} - x^{r_5}); \quad (14)$$

where $x^i$ is called the target vector and $v^i$ is the mutant vector. The variables $x^{r1}, x^{r2}, x^{r3}, x^{r4}$, and $x^{r5}$ are different solutions randomly selected from $P$, which are also different from $x^i$. The scaling factor $F > 0$ controls the impact of the vector differences on the mutant vector. $K \in [0, 1]$ plays a similar role to $F$. For the last two mutation operators, the offspring is the direct output of mutation, where the crossover operator will not be used. For the first two mutation operators, a crossover operator is applied upon $x^i$ and $v_i$ for generating the offspring $u^i$. The binomial crossover is used in our experiments. It works as

$$u_j^i = \begin{cases} v_j^i, & \text{if } rand \leq CR \text{ or } j = j_{rand} \\ x_j^i, & \text{otherwise} \end{cases} \quad (15)$$

where $j \in \{1, \ldots, n\}$ and $rand$ is a uniformly random number from [0, 1]. The crossover rate $CR \in [0, 1]$ is a user-defined control parameter. $j_{rand}$ is an integer randomly chosen from the set $S = 1, \ldots, n$. After the application of these DE operators, a generated offspring might undergo, with a small probability, the polynomial mutation operator.

**Reward Calculation:** In a recent work on AOS for selecting DE mutation operators, a DE mutation operator is rewarded based on the fitness improvement of the offspring compared with its corresponding target vector. In MOEA/D, different solutions correspond to different subproblems, and thus, different comparisons are based on different aggregations of objective functions. In this paper, if an offspring successfully replaces $x^i$, the DE mutation operator that generates it will receive the following reward:

$$\eta = \frac{g(x^i|\lambda^i, z^*) - g(y|\lambda^i, z^*)}{g(x^i|\lambda^i, z^*)} \quad (16)$$

where $\lambda^i$ is the weight vector for subproblem $i$.

An operator $op$ may receive several rewards due to its generated offspring $y$. We sum up all these reward values as its final reward $FIR_{op}$.

## 3.2 Controlling the $\varepsilon$ level

Usually, the $\varepsilon$ level is controlled according to the equation(17). The initial $\varepsilon$ level $\varepsilon(0)$ is the constraint violation of the top $\theta$-th individual in the initial search points. The $\varepsilon$ level is updated until the number of iterations $t$ becomes the control generation $Tc$. After the number of iterations exceeds $Tc$, the $\varepsilon$ level is set to 0 to obtain solutions with minimum constraint violation.

$$\begin{aligned} \varepsilon(0) &= \phi(x_\theta) \\ \varepsilon(t) &= \begin{cases} \varepsilon(0)(1 - \frac{t}{T_c})^{cp}), & 0 < t < T_c \\ 0, & t \geq T_c \end{cases} \end{aligned} \quad (17)$$

where $x_\theta$ is the top $\theta$-th individual. If $\theta > 1$, $\phi(x_\theta) = \theta \times \max_x \phi(x)$.

In this study, a simple scheme of setting the parameter value $cp$ is proposed: The $\varepsilon$ level is adjusted to be a small value $\varepsilon_\lambda = 10^{-5}$ at the generation $T_\lambda = 0.95T_c$.

$$\varepsilon(T_\lambda) = \varepsilon(0)(1 - T_\lambda/T_c) = \varepsilon_\lambda \quad (18)$$

$$cp = (\log \varepsilon - \log \varepsilon(0))/\log(1 - T_\lambda/T_c) \quad (19)$$

$$= (-5 - \log \varepsilon(0))/\log 0.05 \quad (20)$$

To avoid too small value of $cp$, if $cp$ is less than $cp_{\min}$, $cp$ is to be $cp_{\min}$ where $cp_{\min} = 3$. Also, after $T_\lambda$, in order to increase the $\varepsilon$ level, to enlarge $\varepsilon$-feasible region and to search better objective values, $cp$ is decreased and $F$ is increased as follows:

$$cp = 0.3cp + 0.7cp_{\min} \quad (21)$$

$$F = 0.3F_0 + 0.7 \quad (22)$$

## 3.3 Applying the $\varepsilon$ Constrained Method and FRRMAB to MOEA/D

In this paper, we employ the $\varepsilon$ constrained method and FRRMAB into MOEA/D-DE framework. The pseudocode of the whole algorithm is detailed in Algorithm 2.

# 4. EXPERIMENTAL SETTINGS AND PERFORMANCE INDICATOR

## 4.1 Parameters Settings

The parameters settings of our proposed approach are given as follows:

1. $N$: 600 for two objectives and 1000 for three objectives;

2. The number of decision variables of the test instances is set to 10;
3. $T = 0.1N$ and $n_r = 0.01N$;
4. $\delta = 0.9$;
5. In DE and mutation operators: $CR = 1.0$ and $F = 0.5$, $\eta = 20$ and $p_m = 1/n$.
6. Stopping criterion: the algorithm stops after 300,000 function evaluations for each test instance.
7. Control parameters in FRRMAB:
   Scaling factor: $C = 5.0$;
   Size of sliding window: $W = 0.5 \times N$;
   Decaying factor: $D = 1.0$.
8. Control parameters in the $\varepsilon$ level comparison:
   $T_c$: 800 for two objectives and 1000 for three objectives;
   $\theta = 1.5$.

## 4.2 Performance Metric IGD

Let $P^*$ be a set of uniformly distributed points along the PF. Let $A$ be an approximate set to the PF, the average distance from $P^*$ to $A$ is defined as:

$$D(A, P) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \quad (23)$$

where $d(v, A)$ is the minimum Euclidean distance between $v$ and the points in $A$. If $P^*$ is large enough to represent the PF very well, $D(A, P)$ could measure both the diversity and convergence of $A$ in a sense. Ten constrained MOPs (CF1 to CF10) [16] are used in our experimental studies. CMOEA/D-DE-ATP and the three best performers [10], [7], [6] in CEC 2009 MOEA competition are compared with our proposed method. All of them have been run 30 times independently for each test instance.

# 5. EXPERIMENTAL RESULTS AND DISCUSSIONS

Table 1 summarizes the performance of various algorithms in terms of IGD metric in statistics based on 30 independent runs. These state-of-art approaches include our proposed approach (YC) and approach proposed by Jan and Zhang(JZ [4]), and Tseng and Chen(TC [10]), Liu and li (LL [6]), and Liu et. al's (LI [7]).

It is clear to see that our proposed algorithm has the best (lowest) IGD values for CF1, CF6, and CF10. It can also be observed that our proposed approach has better performance than CMOEA/D-DE-ATP [12] on most test instances. This indicates that using FRRMAB and the $\varepsilon$ method to MOEAD-DE is beneficial, compared with penalty function.

# 6. CONCLUSION

In this paper, the $\varepsilon$ constrained method and a bandit-based AOS method FRRMAB are introduced into the MOEA /D-DE framework to tackle constrained multiobjective optimization problems. The experimental results show the proposed algorithm is very competitive on most CF test instances compared with other state-of-art approaches. The results of our proposed algorithm is very preliminary. The future works include further improvement of the performance through adopting advanced learning mechanisms.

**Algorithm 2**:MOEAD-$\varepsilon$DE

**Input:**

1. CMOP(1);
2. a stopping criterion;
3. $N$: the number of subproblems; the population size of $P$ and $A$;
4. a uniform spread of $N$ weight vectors: $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^N$;
5. $T$: the number of the weight vectors in the neighborhood of each weight vector.

**Output:** A set of non-dominated solutions EP;

**Step 1 Initialization:**

a) Compute the Euclidean distances between any two weight vectors and then work out the $T$ closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = i_1, \ldots, i_T$ where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the $T$ closest weight vectors to $\lambda^i$.

b) Generate an initial population $x^1, \ldots, x^N$ by uniformly randomly sampling from the search space.

c) Initialize $z = (z_1, \ldots, z_m)^T$ by setting $z_i = \min\{f_i(x^1), f_i(x^2), \ldots, f_i(x^N)\}$.

d) Set $gen = 1$ and $\varepsilon = \varepsilon(0)$.

**Step 2 Update:**

For each $i = 1, \ldots, N$, do:

**a) Selection of operator:** The operator $op$ is selected according to the **Algorithm 1**.

**b) Selection of Mating/Update Range:** Uniformly randomly generate a number $rand$ from (0,1). If $rand < \delta$, then set $P = B(i)$, otherwise $P = 1, \ldots, N$.

**c) Reproduction:** Randomly select some indexes from $P$, and then generate a solution $y'$ by the application of the chosen DE mutation operator $op$ over the selected solutions; Then apply polynomial mutation operator on $y'$ with probability $p_m$, to produce the offspring $y$;

**d) Repair:** If an element of $y$ is out of the boundary, its value is reset to be randomly selected value inside the boundary.

**e) Evaluate** $y$ **:** $FV^y = F(y), V(y) = |\sum_{j=1}^{p} \min(g_j(y), 0)|$.

**f) Update of** $z$ **:** For each $j = 1, \ldots, m$,if $z_j > f_j(y)$, then set $z_j = f_j(y)$.

**g) Update of Neighboring Solutions:** For each index $j \in P$, if $(g^{te}(y|\lambda^j, z), \phi(y)) <_\varepsilon (g^{te}(x^j|\lambda^j, z), \phi(x^j))$, then set $x^j = y$ and $FV^j = F(y)$.

**h) Evaluate** $FIR_{op}$**:** Sum up the rewards that are generated by the update of $y$ as the $FIR_{op}$ value.

**i) Update of FIFO queue:** The node which is combined operator $op$ and $FRR$ value is added at the tail of the FIFO queue. Then, the FRR values are evaluated for each operator according to the FIFO queue.

**j) Update of EP:** Remove from EP all the vectors dominated by $F(y')$. Add to EP if no vectors in EP dominate $F(y')$.

**Step 3: Stopping Criteria:** If stopping criteria is satisfied, then stop and output EP.

**Step 4: Control the** $\varepsilon$ **level:** $\varepsilon = \varepsilon(gen)$, $\varepsilon(gen)$ is described in equation (17).

**Step 5:** Set $gen = gen + 1$. Go to **Step 2**.

Table 1: The Comparison of Various Algorithms in IGD. The results in **boldface** indicate the best performance

| | Instance | CF1 | CF2 | CF3 | CF4 | CF5 | CF6 | CF7 | CF8 | CF9 | CF10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| best | YC | **0.000021** | 0.0026 | 0.0738 | 0.0077 | 0.031 | **0.0059** | 0.0343 | 0.0678 | 0.0455 | **0.0973** |
| | JZ | 0.000314 | 0.002568 | 0.052966 | **0.005301** | 0.060908 | 0.007778 | 0.048766 | 0.070507 | **0.039339** | 0.142689 |
| | TC | 0.013854 | 0.004142 | 0.0753 | 0.008937 | 0.017565 | 0.009568 | 0.018691 | 0.621968 | 0.072071 | 0.117317 |
| | LL | 0.000682 | 0.002733 | 0.090844 | 0.008964 | 0.058818 | 0.009022 | 0.05351 | 0.047328 | 0.046035 | 0.105482 |
| | LI | 0.007061 | **0.001579** | **0.038062** | 0.005523 | **0.007872** | 0.006186 | **0.010424** | **0.038785** | 0.119107 | 0.098377 |
| worst | YC | **0.000066** | 0.026 | 0.2723 | 0.0134 | 0.2789 | **0.0132** | 0.2007 | 0.0827 | **0.0536** | 0.521 |
| | JZ | 0.001102 | 0.089054 | 0.387998 | 0.128202 | 0.47448 | 0.061374 | 0.525922 | 0.111194 | 0.061451 | 1.273927 |
| | TC | 0.023597 | 0.051815 | 0.142828 | 0.014276 | **0.027832** | 0.038112 | 0.037144 | 1.42867 | 0.096282 | **0.163473** |
| | LL | 0.001147 | 0.013135 | 0.251884 | 0.023999 | 0.192996 | 0.019939 | 0.203867 | 0.098487 | 0.058389 | 0.416232 |
| | LI | 0.016932 | **0.003063** | **0.070731** | **0.011552** | 0.039396 | 0.03112 | **0.033821** | **0.065014** | 0.206549 | 0.239399 |
| mean | YC | **0.000040** | 0.005 | 0.1771 | 0.0106 | 0.1247 | **0.008** | 0.0684 | 0.0759 | 0.0499 | 0.2309 |
| | JZ | 0.00054 | 0.009402 | 0.185618 | 0.015356 | 0.227288 | 0.034201 | 0.200548 | 0.086989 | **0.04897** | 0.325308 |
| | TC | 0.019187 | 0.026779 | 0.10446 | 0.011096 | 0.020779 | 0.016168 | 0.024695 | 1.08544 | 0.085139 | **0.137648** |
| | LL | 0.000859 | 0.004203 | 0.182905 | 0.014232 | 0.10973 | 0.013948 | 0.10446 | 0.060746 | 0.050549 | 0.197409 |
| | LI | 0.011311 | **0.0021** | **0.056305** | **0.006995** | **0.015773** | 0.01502 | **0.019051** | **0.047501** | 0.143432 | 0.162128 |
| st.dev. | YC | **0.000012** | 0.0041 | 0.0363 | 0.0014 | 0.0836 | **0.002** | 0.0406 | **0.0035** | **0.0018** | 0.1452 |
| | JZ | 0.000184 | 0.021539 | 0.079894 | 0.023468 | 0.116536 | 0.015884 | 0.115529 | 0.009889 | 0.006329 | 0.290022 |
| | TC | 0.002568 | 0.014715 | 0.015595 | **0.001368** | **0.002421** | 0.005985 | **0.004654** | 0.219108 | 0.008191 | **0.009215** |
| | LL | 0.00011 | 0.002635 | 0.042127 | 0.003293 | 0.030676 | 0.002586 | 0.035116 | 0.012972 | 0.003357 | 0.076004 |
| | LI | 0.002758 | **0.000453** | **0.007573** | 0.001457 | 0.006662 | 0.006462 | 0.006123 | 0.006387 | 0.021416 | 0.031621 |

# 7. REFERENCES


[1] *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, 18-21 May, 2009*. IEEE, 2009.

[2] C. A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002.

[3] Á. Fialho. Adaptive operator selection for optimization. *PhD, Ecole Doctorale d'Informatique, Universite Paris-Sud, Paris*, 2010.

[4] M. A. Jan and Q. Zhang. Eee moea/d for constrained multiobjective optimization: Some preliminary experimental results. *UKCI*, 2010.

[5] K. Li, Á. Fialho, S. Kwong, and Q. Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolutionary Computation*, 18(1):114–130, 2014.

[6] H. lin Liu and X. Li. The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In *IEEE Congress on Evolutionary Computation* [1], pages 1928–1934.

[7] M. Liu, X. Zou, Y. Chen, and Z. Wu. Performance assessment of dmoea-dd with cec 2009 moea competition test instances. In *IEEE Congress on Evolutionary Computation* [1], pages 2913–2918.

[8] T. Takahama and S. Sakai. Constrained optimization by applying the $\alpha$ constrained method to the nonlinear simplex method with mutations. *Evolutionary Computation, IEEE Transactions on*, 9(5):437–451, 2005.

[9] T. Takahama and S. Sakai. Efficient constrained optimization by the constrained differential evolution with rough approximation using kernel regression. In *IEEE Congress on Evolutionary Computation*, pages 1334–1341. IEEE, 2013.

[10] L.-Y. Tseng and C. Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *IEEE Congress on Evolutionary Computation* [1], pages 1951–1958.

[11] J. Yao, Q. Zhang, and J. Lei. Recent developments in natural computation. *Neurocomputing*, 72(13-15):2833–2834, 2009.

[12] Q. Zhang et al. Moea/d for constrained multiobjective optimization: Some preliminary experimental results. In *2010 UK Workshop on Computational Intelligence (UKCI)*, pages 1–6, 2010.

[13] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolutionary Computation*, 11(6):712–731, 2007.

[14] Q. Zhang, W. Liu, and H. Li. The performance of a new version of moea/d on cec09 unconstrained mop test instances. In *IEEE Congress on Evolutionary Computation* [1], pages 203–208.

[15] Q. Zhang, W. Liu, and H. Li. The performance of a new version of moea/d on cec09 unconstrained mop test instances. In *IEEE Congress on Evolutionary Computation* [1], pages 203–208.

[16] 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. *University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report*, 2008.