H. Ishibuchi, R. Imada, Y. Setoguchi and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," *Proc. of 2016 IEEE Congress on Evolutionary Computation (Accepted)*.

Performance Comparison of NSGA-II and NSGA-III on Various Many-Objective Test Problems

Hisao Ishibuchi, Ryo Imada, Yu Setoguchi and Yusuke Nojima
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
{hisaoi@, ryo.imada@ci., yu.setoguchi@ci., nojima@}cs.osakafu-u.ac.jp

***Abstract*—Recently NSGA-III has been frequently used for performance comparison of newly proposed evolutionary many-objective optimization algorithms. That is, NSGA-III has been used as a benchmark algorithm for evolutionary many-objective optimization. However, unfortunately, its source code is not available from the authors of the NSGA-III paper. This leads to an undesirable situation where a different implementation is used in a different study. Moreover, comparison is usually performed on DTLZ and WFG test problems. As a result, the performance of NSGA-III on a wide variety of many-objective test problems is still unclear whereas it has been frequently used for performance comparison in the literature. In this paper, we evaluate the performance of NSGA-III in comparison with NSGA-II on four totally different types of many-objective test problems with 3-10 objectives: DTLZ1-4 problems, their maximization variants, distance minimization problems, and knapsack problems. We use two different implementations of NSGA-II and NSGA-III. We show through computational experiments that NSGA-III does not always outperform NSGA-II even for ten-objective problems. That is, their comparison results depend not only on the number of objectives but also on the type of test problems. The choice of test problems has a larger effect on their comparison results than the number of objectives in our computational experiments. We also demonstrate that totally different results are obtained from different implementations of NSGA-III for some test problems.**

Keywords*—*Evolutionary multiobjective optimization (EMO), evolutionary many-objective optimization, many-objective problems, NSGA-II, NSGA-III.

I. Introduction

NSGA-II has been the most frequently-used evolutionary multiobjective optimization (EMO) algorithm in the literature since its proposal [1]-[3]. New EMO algorithms were almost always compared with NSGA-II for performance evaluation in 2000-2010. However, it is well-known that Pareto dominance-based EMO algorithms such as NSGA-II and SPEA [4] do not work well on many-objective test problems [5]-[7]. Recently, NSGA-III [8] was proposed as an evolutionary many-objective algorithm. NSGA-III has already been playing the same role as NSGA-II: Newly proposed many-objective algorithms have been compared with NSGA-III for performance evaluation. For example, high performance of reference direction-based many-objective algorithms such as I-DBEA [9], MOEA/DD [10] and θ-DEA [11] was demonstrated in comparison with NSGA-III on many-objective DTLZ [12] and WFG [13] test problems.

This work was supported by JSPS KAKENHI Grant Numbers 24300090 and 26540128.

Newly proposed evolutionary many-objective algorithms are usually evaluated by computational experiments on the DTLZ and WFG test problems in the literature. Since these test problems were designed using a similar mechanism [14], we may have to say that even the most frequently-used many-objective algorithm (i.e., NSGA-III) has not been evaluated on a wide variety of test problems. Moreover, many-objective problems are not always difficult for classical EMO algorithms even when they have a large number of objectives [15], [16]. Thus it may be possible that NSGA-III does not outperform classical EMO algorithms such as NSGA-II and SPEA in their applications to some other many-objective test problems.

In this paper, we examine the performance of NSGA-III in comparison with NSGA-II using four, totally different, types of test problems with 3-10 objectives: DTLZ1-4 problems [12], their maximization variants where all objectives are maximized (while they are minimized in the original DTLZ1-4 problems), distance minimization problems with 10-1000 variables [17], and knapsack problems with some correlated objectives [16].

Unfortunately there is no standard source code of NSGA-III since it is not available from the authors of the NSGA-III paper [8]. The current situation seems to be as follows: a different implementation of NSGA-III is used in a different study on evolutionary many-objective optimization. Recently NSGA-III has been included in the revised jMetal website (the development site of jMetal 5.0 [18]). In this paper, we use the NSGA-III algorithm in jMetal 5.0 and another downloadable implementation [19] by the authors of the θ-DEA paper [11]. The use of the different implementations is for examining the difference of NSGA-III between them. With respect to NSGA-II, we use the jMetal code since we use the jMetal code of NSGA-III. For comparison, we also use our implementation of NSGA-II.

This paper is organized as follows. In Section II, we briefly explain our test problems used in computational experiments: DTLZ1-4, their maximization variants, distance minimization problems, and knapsack problems. In Section III, we report performance comparison results between NSGA-II and NSGA-III. Two implementations are examined for each algorithm. In Section IV, we examine the behavior of each algorithm in detail for discussing why NSGA-II outperforms NSGA-III on some many-objective test problems even when they have ten objectives. Finally, we conclude this paper in Section V.

II. Many-Objective Test Problems

A. DTLZ Problems and their Maximization Variants

DTLZ [12] is a set of scalable multiobjective minimization problems where the number of objectives (i.e., m) can be arbitrarily specified. Especially, DTLZ1-4 test problems have been frequently used in the literature.

The objective functions of DTLZ1-4 have the following special structure [12]:

$$f_j(\boldsymbol{x}) = (1 + g(\boldsymbol{x}_M))h_j(\boldsymbol{x}_{\text{pos}}),\ j = 1, 2, ..., m, \qquad (1)$$

where $h_j(\boldsymbol{x}_{\text{pos}})$ is a function of $(m-1)$ variables in $\boldsymbol{x}_{\text{pos}}$, and $g(\boldsymbol{x}_M)$ is a function of k variables in $\boldsymbol{x}_M$. Whereas the value of k can be arbitrarily specified, the total number of variables in $\boldsymbol{x}_{\text{pos}}$ and $\boldsymbol{x}_M$ is usually small in the literature. In this paper, we specify k as $k = 5$ in DTLZ1 and $k = 10$ in DTLZ2-4. Thus the total number of variables in each m-objective test problem is $(5+m-1)$ in DTLZ1 and $(10+m-1)$ in DTLZ2-4. These specifications have been frequently used in the literature.

In Fig. 1, we show the feasible region of each test problem for the case of $m = 2$ (i.e., two objectives). We also show the Pareto front (a red curve) of each test problem and randomly generated 100 solutions (open circles). Except for DTLZ4, a set of the randomly generated solutions in Fig. 1 has a much larger diversity than the Pareto front of the corresponding problem (especially DTLZ1 and DTLZ3).

The maximization variants of DTLZ1-4 are exactly the same as DTLZ1-4 except for the maximization of all objectives (they are minimized in the original DTLZ1-4 problems). The Pareto front of each variant is shown by a blue curve in Fig. 1. In this paper, the maximization variants of DTLZ1-4 are referred to as Max-DTLZ1-4 (see Fig. 1).

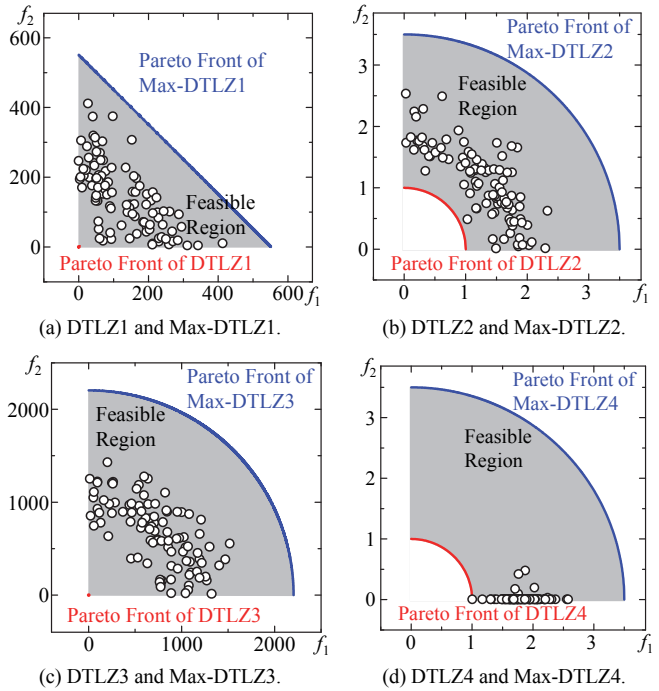
Each DTLZ problem and its maximization variant have exactly the same feasible region (see Fig. 1). However, the Pareto front of a maximization variant is much wider than that of its original problem. Thus we need a strong diversification mechanism in EMO algorithms when we try to approximate the entire Pareto front of each maximization variant.

We use the hypervolume for performance comparison in our computational experiments. For the original DTLZ1-4 problems, the reference point for hypervolume calculation is specified as (0.6, 0.6, ..., 0.6) for DTLZ1 and (1.1, 1.1, ..., 1.1) for DTLZ2-4. For their maximization variants, the reference point is specified as (−50.0, −50.0, ..., −50.0) for Max-DTLZ1 and Max-DTLZ3, and (−0.5, −0.5, ..., −0.5) for Max-DTLZ2 and Max-DTLZ4.

B. Distance Minimization Problems

In [20], two-dimensional distance minimization problems with four equivalent Pareto regions were used to visually examine the diversity of solutions in the decision space. We generate test problems (2D distance minimization problems) in the same manner as in [20] on the two-dimensional decision space $[-50, 50] \times [-50, 50]$ in Fig. 2.

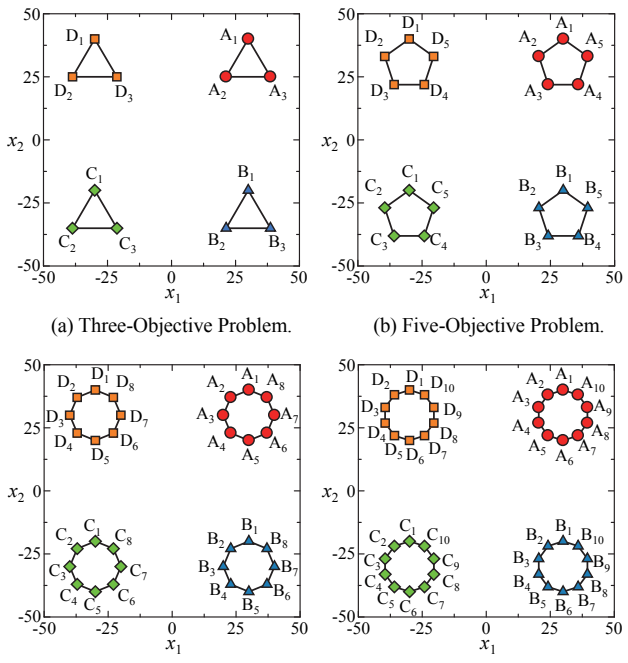
Each problem has four polygons of the same shape and the same size. The ith objective is to minimize the distance to the nearest ith vertex among the four ith vertexes. For example, the first objective is to minimize the distance to the nearest first vertex among A_1, B_1, C_1 and D_1. The number of objectives is the same as the number of the vertexes in each polygon. All points inside each polygon (including points on the sides and the vertexes) are Pareto optimal solutions in the decision space. The reference point for hypervolume calculation is specified as (20, 20, ..., 20) in the objective space where 20 is the maximum distance between two vertexes in each polygon in Fig. 2.



(a) DTLZ1 and Max-DTLZ1. (b) DTLZ2 and Max-DTLZ2.

(c) DTLZ3 and Max-DTLZ3. (d) DTLZ4 and Max-DTLZ4.

Fig. 1. Explanation of the two-objective DTLZ and Max-DTLZ problems.



(a) Three-Objective Problem. (b) Five-Objective Problem.

(c) Eight-Objective Problem. (d) Ten-Objective Problem.

Fig. 2. Two-dimensional distance minimization problems.

The two-dimensional decision space in [20] was extended to a general d-dimensional decision space $[-50, 50]^d$ in [17]. In this paper, we use 10D ($d=10$), 100D ($d=100$) and 1000D ($d=1000$) problems. Each test problem is defined by a single polygon as shown in Fig. 3. Each polygon in Fig. 3 is the projection of the corresponding polygon in the decision space $[-50, 50]^d$ to its two-dimensional subspace. The ith objective is to minimize the distance to the ith vertex, which is a point in the d-dimensional decision space. Each polygon in Fig. 3 is placed on a hyperplane. All points inside the polygon on the hyperplane are Pareto optimal solutions. For details of the test problem design, see [17]. The reference point for hypervolume calculation is specified as (112, 112, ..., 112) for the 10D problems, (354, 354, ..., 354) for the 100D problems, and (1119, 1119, ..., 1119) for the 1000D problems using the maximum distance between two vertexes of the same polygon.

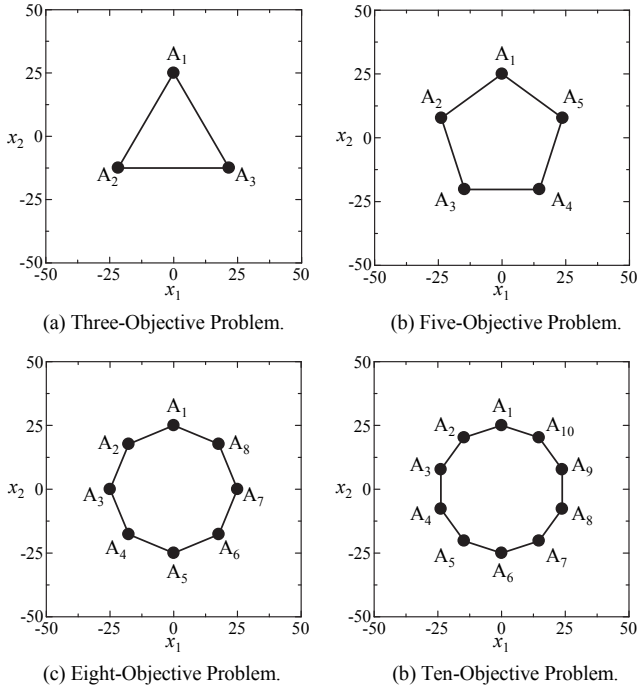


Fig. 3. High-dimensional distance minimization problems.

C. Knapsack Problems

In the same manner as [16], we generate ten-objective 500-item knapsack problems with correlated objectives. The first two objectives $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are the same as those of the two-objective 500-item knapsack problem in Zitzler & Thiele [4]. The constraint conditions are also the same as those in [4]. The other eight objectives are generated in the following manner. First, they are generated by randomly specifying the profit a_{ij} of item j for objective i as an integer in the interval [10, 100]:

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{500} a_{ij} x_j \,, \quad i = 3, 4, ..., 10. \tag{2}$$

Then, using a real number parameter α in [0, 1], we generate correlated objectives of our ten-objective 500-item knapsack problem as follows:

$$g_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) \,, \quad i = 1, 2, \tag{3}$$

$$g_i(\boldsymbol{x}) = \alpha \cdot f_1(\boldsymbol{x}) + (1-\alpha) \cdot f_i(\boldsymbol{x}) \,, \quad i = 3, 5, 7, 9, \tag{4}$$

$$g_i(\boldsymbol{x}) = \alpha \cdot f_2(\boldsymbol{x}) + (1-\alpha) \cdot f_i(\boldsymbol{x}) \,, \quad i = 4, 6, 8, 10. \tag{5}$$

The parameter α in [0, 1] can be viewed as a kind of correlation strength. When α = 1 (i.e., maximum correlation), $g_i(\boldsymbol{x})$ is the same as $f_1(\boldsymbol{x})$ or $f_2(\boldsymbol{x})$. When α = 0 (i.e., minimum correlation), $g_i(\boldsymbol{x})$ is the same as the randomly generated $f_i(\boldsymbol{x})$. In this paper, we examine its four values: $\alpha = 0.2, 0.4, 0.6, 0.8$. Our knapsack problems with three, five and eight objectives are generated by using the first three, five and eight objectives from the generated ten objectives $g_i(\boldsymbol{x})$, i = 1, 2, ..., 10.

In Fig. 4, we show the Pareto front and randomly generated 100 solutions for the two-objective 500-item knapsack problem. The length of the Pareto front is much larger than the spread of randomly generated solutions. Thus a strong diversification mechanism is needed for finding a well distributed solutions over the Pareto front. At the same time, a strong convergence mechanism is also needed to push the population of randomly generated solutions to the Pareto front. In our computational experiments, the origin of the objective space is used as the reference point for hypervolume calculation.

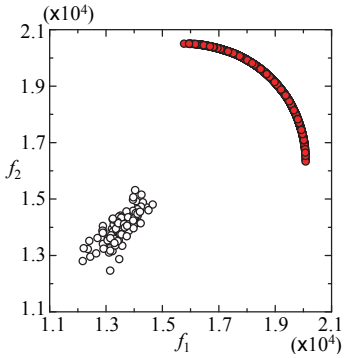


Fig. 4. Pareto front (red circles) and randomly generated 100 solutions (open circles) for the two-objective 500-item knapsack problems.

III. Computational Experiments

A. Examined Algorithms: NSGA-II and NSGA-III

NSGA-II [1]-[3] is a well-known and frequently-used EMO algorithm. It has a $(\mu+\lambda)$ES-style generation update structure. Usually μ and λ are the same: $\mu=\lambda$. We use this specification in this paper. Solutions in a population are classified into different groups (i.e., they are sorted into different ranks) by Pareto dominance relation among solutions. Then the crowding distance is calculated for each solution using the solutions in each group. We use the jMetal code of NSGA-II. We also use our own implementation. In general, algorithm implementation needs a lot of minor unwritten specifications. The following are examples of minor specifications in our implementation.

1. Handling of Overlapping Solutions in the Objective Space:

All overlapping solutions with the same objective vector are classified into the same group. Different numbers are assigned to those solutions for always sorting them in the same order with respect to each objective in the crowding distance calculation. For example, when four solutions ($\boldsymbol{x}_A$, $\boldsymbol{x}_B$, $\boldsymbol{x}_C$, $\boldsymbol{x}_D$) are overlapping, they are always sorted in the same order (e.g.,

an alphabetical order: $\boldsymbol{x}_A$, $\boldsymbol{x}_B$, $\boldsymbol{x}_C$, $\boldsymbol{x}_D$). In this case, the crowding distances of $\boldsymbol{x}_B$ and $\boldsymbol{x}_C$ are always calculated as zero. As a result, $\boldsymbol{x}_B$ and $\boldsymbol{x}_C$ are likely to be removed in the generation update phase. If they are randomly sorted for each objective, all the four solutions can have large values of the crowding distance. As a result, they will be able to survive for many generations.

2. Handling of Overlapping Solutions on Each Objective:

Let us assume that all solutions in the same group (with the same rank) have the same value of an objective. In this case, first they are ordered. Then an infinitely large value is assigned to the first and last solutions as the distance on this objective. The distance zero is assigned to all the other solutions. The normalization procedure is not applied to the assigned distance values zero and infinity.

3. Handling of Solutions with the Same Fitness:

Solutions with the same crowding distance in the same group (with the same rank) are always handled randomly in the generation update (i.e., environmental selection) and the parent selection (i.e., mating selection).

4. Recalculation of Crowding Distance:

Before the generation update, the crowding distance of each solution is calculated in the merged population with $(\mu+\mu)$ solutions. After the generation update, the crowding distance of each solution in the new population of size μ is not always the same as the calculated value before the generation update. We recalculate the crowding distance of all solutions in the worst group with the worst rank in the new population.

These minor specifications do not have large effects on the performance of NSGA-II. As shown later in this section, we do not observe any large differences in the performance of NSGA-II between the jMetal code and our own implementation.

NSGA-III [8], which was designed for many-objective optimization, has also a $(\mu+\lambda)$ES-style generation update structure $(\mu=\lambda)$. As in NSGA-II, Pareto dominance relation among solutions in a population is used for fitness evaluation. However, the crowding distance is not used in NSGA-III. This is because a good distribution of solutions over a high-dimensional Pareto front cannot be maintained by the distance calculation on each objective. In NSGA-III, a set of uniformly distributed direction vectors is used for diversity maintenance. The basic idea is to find a non-dominated solution around the intersection of the Pareto front and each reference line.

As we have already explained for NSGA-II, a lot of minor specifications are usually needed for implementing any EMO algorithm. Most of those specifications are not clearly written in the original paper where the algorithm was proposed. Since the source code of NSGA-III is not available from the authors of the NSGA-III paper, there exist a number of unknown minor specifications. Thus it is likely that we will obtain different performance evaluation results from different implementations of NSGA-III. In this paper, we use the jMetal code [18] and the implementation by the authors of the θ-DEA paper [11] downloadable from [19].

B. Parameter Specifications

The population size and the termination condition for NSGA-II and NSGA-III are specified for all test problems as shown in Table I. For the continuous test problems (i.e., DTLZ, Max-DTLZ, distance minimization), we use the SBX crossover with the distribution index 20 and the polynomial mutation with the distribution index 30. The crossover probability is specified as 1.0, and the mutation probability is specified as $1/n$ where n is the string length. For the knapsack problems with binary decision variables, we use the uniform crossover with the crossover probability 1.0 and the bit-flip mutation with the mutation probability 1/500 where 500 is the string length.

TABLE I. POPULATION SIZE AND TERMINATION CONDITION.

Objectives	Population Size	Termination Condition (Number of Examined Solutions)
$m = 3$	92	92 x 300 = 27,600
$m = 5$	212	212 x 500 = 106,000
$m = 8$	156	156 x 800 = 124,800
$m = 10$	276	276 x 1000 = 276,000

C. Performance Comparison Results

Average hypervolume values over 11 runs are summarized in Tables II-V in the next page for each test problem with 3, 5, 8 and 10 objectives, respectively. In each table, the best and the second best average hypervolume value for each test problem are highlighted by yellow and light yellow, respectively. The value "0" in the tables means that the hypervolume value is zero in all of the 11 runs. That is, no solution that dominates the reference point is obtained in those runs.

Experimental results on the DTLZ1-4 test problems clearly show that NSGA-II does not work well on many-objective problems with eight objectives in Table IV and ten objectives in Table V. That is, we can observe the severe deterioration of the convergence performance of NSGA-II by the increase in the number of objectives. For the distance minimization test problems with five or more objectives, we can also observe that NSGA-III clearly outperforms NSGA-II in Tables III-V. A common feature of these two types of test problems is that the size of the Pareto front is relatively small in comparison with the size of the feasible region in the objective or decision space (see Figs. 1-3). Thus the convergence of solutions to a small region is important for obtaining good experimental results.

However, we cannot observe such a clear performance deterioration of NSGA-II by the increase in the number of objectives for the Max-DTLZ1-4 test problems in comparison with NSGA-III. That is, the effects of the number of objectives on the performance of NSGA-II and NSGA-III are unclear for the Max-DTLZ1-4 test problems. For the knapsack problems with eight and ten objectives, NSGA-II outperforms NSGA-III in Tables IV and Table V. A common feature of these two types of test problems is the importance of the diversification of solutions since the size of the Pareto front of each problem is large (see Fig. 1 and Fig. 4).

Our experimental results demonstrate that the choice of test problems has a dominant effect on performance comparison results between NSGA-II and NSGA-III. Whereas NSGA-III clearly outperforms NSGA-II when the DTLZ1-4 problems are used as in many studies in the literature, its superiority is not clear on their maximization variants. Moreover, NSGA-II outperforms NSGA-III on many-objective knapsack problems.

TABLE II. AVERAGE RESULTS ON THREE-OBJECTIVE PROBLEMS.

Test Problems	NSGA-II		NSGA-III	
	jMetal 5.0	Our Lab	jMetal 5.0	Yuan [19]
DTLZ1	1.250E–01	1.657E–01	1.865E–01	1.889E–01
DTLZ2	7.009E–01	7.013E–01	7.442E–01	7.444E–01
DTLZ3	0	0	0	2.542E–01
DTLZ4	7.060E–01	7.018E–01	6.611E–01	7.443E–01
Max-DTLZ1	4.196E+07	4.227E+07	2.068E+07	4.250E+07
Max-DTLZ2	3.453E+01	3.471E+01	2.935E+01	3.508E+01
Max-DTLZ3	5.004E+09	5.081E+09	4.546E+09	5.123E+09
Max-DTLZ4	3.483E+01	3.465E+01	3.116E+01	3.522E+01
Distance-2D	2.516E+03	2.517E+03	2.498E+03	2.499E+03
Distance-10D	3.816E+05	3.836E+05	3.809E+05	3.819E+05
Distance-100D	6.845E+06	6.841E+06	6.903E+06	7.177E+06
Distance-1000D	5.415E+07	5.678E+07	7.263E+07	7.723E+07
Knapsack (0.2)	6.418E+12	6.312E+12	6.329E+12	6.523E+12
Knapsack (0.4)	6.457E+12	6.398E+12	6.400E+12	6.572E+12
Knapsack (0.6)	6.545E+12	6.533E+12	6.522E+12	6.721E+12
Knapsack (0.8)	6.745E+12	6.708E+12	6.703E+12	6.930E+12

TABLE III. AVERAGE RESULTS ON FIVE-OBJECTIVE PROBLEMS.

Test Problems	NSGA-II		NSGA-III	
	jMetal 5.0	Our Lab	jMetal 5.0	Yuan [19]
DTLZ1	0	0	7.671E–02	7.662E–02
DTLZ2	0.745E+00	0.728E+00	1.307E+00	1.307E+00
DTLZ3	0	0	0.993E+00	1.097E+00
DTLZ4	0.961E+00	0.041E+00	1.281E+00	1.308E+00
Max-DTLZ1	6.422E+11	6.418E+11	2.237E+11	5.226E+11
Max-DTLZ2	1.337E+02	1.342E+02	0.647E+02	1.157E+02
Max-DTLZ3	3.702E+15	3.678E+15	1.898E+15	2.426E+15
Max-DTLZ4	1.325E+02	1.330E+02	0.885E+02	1.042E+02
Distance-2D	3.080E+05	3.073E+05	3.190E+05	3.170E+05
Distance-10D	1.150E+09	1.134E+09	1.459E+09	1.462E+09
Distance-100D	2.480E+11	2.252E+11	2.661E+11	2.915E+11
Distance-1000D	2.526E+13	2.428E+13	3.565E+13	3.808E+13
Knapsack (0.2)	2.151E+21	2.042E+21	2.025E+21	2.094E+21
Knapsack (0.4)	2.238E+21	2.131E+21	2.099E+21	2.159E+21
Knapsack (0.6)	2.352E+21	2.257E+21	2.246E+21	2.296E+21
Knapsack (0.8)	2.482E+21	2.442E+21	2.398E+21	2.483E+21

TABLE IV. AVERAGE RESULTS ON EIGHT-OBJECTIVE PROBLEMS.

Test Problems	NSGA-II		NSGA-III	
	jMetal 5.0	Our Lab	jMetal 5.0	Yuan [19]
DTLZ1	0	0	1.600E–02	1.677E–02
DTLZ2	0	0	1.877E+00	1.976E+00
DTLZ3	0	0	0.350E+00	1.891E+00
DTLZ4	0	0	1.979E+00	1.980E+00
Max-DTLZ1	1.982E+17	2.041E+17	0.600E+17	3.201E+17
Max-DTLZ2	2.538E+02	2.618E+02	0.551E+02	2.350E+02
Max-DTLZ3	2.593E+23	2.707E+23	0.752E+23	2.752E+23
Max-DTLZ4	2.191E+02	2.267E+02	0.264E+02	2.488E+02
Distance-2D	3.249E+08	3.179E+08	4.005E+08	4.006E+08
Distance-10D	1.103E+14	1.156E+14	2.688E+14	2.891E+14
Distance-100D	0.681E+18	0.582E+18	1.542E+18	1.541E+18
Distance-1000D	1.619E+21	1.129E+21	5.005E+21	5.544E+21
Knapsack (0.2)	1.115E+34	1.063E+34	1.044E+34	1.059E+34
Knapsack (0.4)	1.238E+34	1.193E+34	1.137E+34	1.152E+34
Knapsack (0.6)	1.443E+34	1.372E+34	1.313E+34	1.371E+34
Knapsack (0.8)	1.688E+34	1.624E+34	1.509E+34	1.605E+34

TABLE V. AVERAGE RESULTS ON TEN-OBJECTIVE PROBLEMS.

Test Problems	NSGA-II		NSGA-III	
	jMetal 5.0	Our Lab	jMetal 5.0	Yuan [19]
DTLZ1	0	0	5.843E–03	6.046E–03
DTLZ2	0	0	2.094E+00	2.512E+00
DTLZ3	0	0	1.945E+00	2.498E+00
DTLZ4	0	0	2.515E+00	2.515E+00
Max-DTLZ1	1.050E+21	1.121E+21	0.448E+21	1.896E+21
Max-DTLZ2	3.843E+02	3.851E+02	0.688E+02	5.093E+02
Max-DTLZ3	0.497E+29	0.513E+29	0.062E+29	1.213E+29
Max-DTLZ4	3.123E+02	3.151E+02	0.118E+02	4.885E+02
Distance-2D	4.037E+10	3.953E+10	4.933E+10	4.886E+10
Distance-10D	0.444E+18	0.462E+18	0.924E+18	1.109E+18
Distance-100D	1.843E+22	1.354E+22	4.391E+22	7.038E+22
Distance-1000D	1.051E+27	0.690E+27	2.782E+27	2.461E+27
Knapsack (0.2)	3.565E+42	3.420E+42	3.326E+42	3.373E+42
Knapsack (0.4)	4.103E+42	3.924E+42	3.689E+42	3.703E+42
Knapsack (0.6)	5.134E+42	4.775E+42	4.526E+42	4.511E+42
Knapsack (0.8)	6.457E+42	5.948E+42	5.972E+42	6.046E+42

Now, let us examine the difference in the performance between the two implementations of NSGA-II. In Tables II-V, we cannot find any large differences between them except for the results on the five-objective DTLZ4 problem and the 1000-dimensional distance minimization problems with eight and ten objectives. These test problems are very difficult for NSGA-II (much better results are obtained from NSGA-III). Thus the experimental result of each run of NSGA-II heavily depends on an initial population. As a result, the difference between the two implementation looks large. Except for these three test problems (among the 64 test problems in Tables II-V), similar results are obtained by the two implementations of NSGA-II.

Next, let us examine the difference between the two implementations of NSGA-III. For the DTLZ1-4 problems, it seems that similar results are obtained from them. Differences in the experimental results between the two implementations are small for the distance minimization and knapsack problems, either. However, for the maximization DTLZ1-4 problems, clearly different results are obtained. Moreover, clearly better results are always obtained by Yuan's implementation for the maximization DTLZ1-4 problems in Tables II-V (i.e., for all the 16 test problems in total). It is not likely that those consistent experimental results happen to be obtained from the same algorithm.

As explained for NSGA-II, almost all algorithms have a lot of minor unwritten specifications. So, it is difficult to identify the reason for the difference between the two implementations of NSGA-III. After a long careful check of each code again and again, we find a possible reason. In Yuan's implementation, the normalization is performed using Eq.(4) of the NSGA-III paper [8] as $f_i^n(\boldsymbol{x}) = f_i'(\boldsymbol{x})/a_i$ for each objective i. However, in the jMetal implementation, the normalization is based on $f_i^n(\boldsymbol{x}) = f_i'(\boldsymbol{x})/(a_i - z_i^{\min})$. This was Eq.(5) in the early access version of [8], which has been changed to Eq.(4) in [8].

For comparison, we modify the jMetal code using Eq.(4) in [8] instead of Eq.(5) of its early access version. Then we apply each implementation (i.e., jMetal, modified jMetal, and Yuan [19]) to the ten-objective Max-DTLZ2 problems 100 times. The histogram of the obtained 100 hypervolume values from each implementation is shown in Fig. 5. We can see from Fig. 5 that (i) there is a large difference between the jMetal and Yuan implementations, and (ii) similar results are obtained from the modified jMetal and Yuan implementations. However, it seems that they still have some small difference.

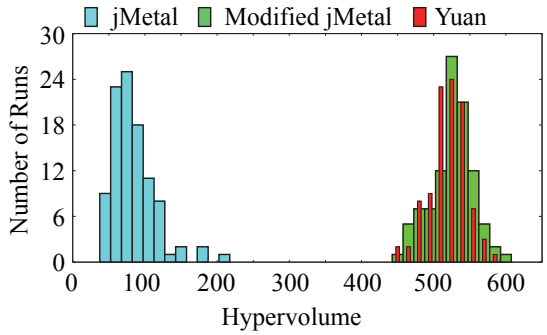
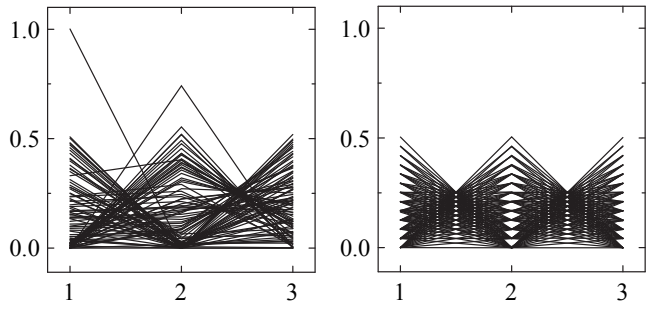


Fig. 5. Histogram of the obtained 100 hypervolume values.

IV. Discussions on Experimental Results

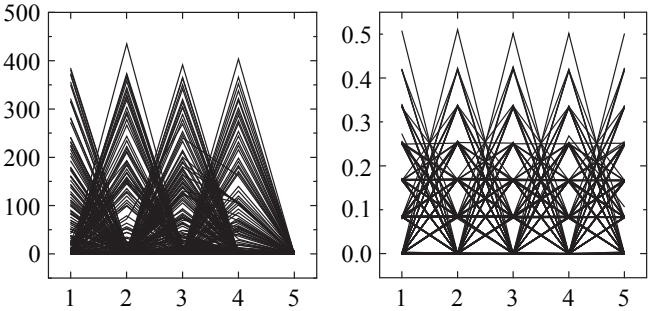
Tables II-V suggest that NSGA-II has high diversification ability while NSGA-III has high convergence ability. In this section, we further examine this observation by showing an obtained solution set by a single run of each algorithm. Among the 11 runs of each algorithm on each test problem, we choose a single run with the median hypervolume value. In Figs. 6-9, we show the obtained solution sets by NSGA-II (jMetal) and NSGA-III (Yuan [19]) on the DTLZ1 problems with three, five, eight and ten objectives, respectively. In this section, the obtained solution set by NSGA-II (NSGA-III) is always shown in the left (right) plot in each figure. Obtained solution sets are shown using the parallel coordinates. The horizontal axis of each graph shows the index of the objective, and the vertical axis shows the corresponding objective value.

In Fig. 6 on the three-objective DTLZ1 problem, similar solution sets are obtained by the two algorithms. However, totally different solution sets are obtained in Figs. 7-9 on the many-objective DTLZ1 problems. Note that the scale of the vertical axis in each left graph in Figs. 7-9 by NSGA-II is about 1000 times larger than that in each right graph. Since the Pareto front of the m-objective DTLZ1 problem is included in $[0, 0.5]^m$, Figs. 7-9 show poor and high convergence ability of NSGA-II and NSGA-III, respectively. Similar results are also obtained for the DTLZ2-4 problems (the results are not shown).



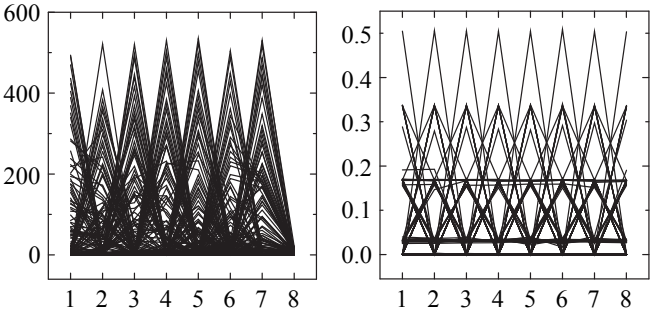
(a) NSGA-II. (b) NSGA-III.

Fig. 6. Obtained solution sets on the three-objective DTLZ1 problem.



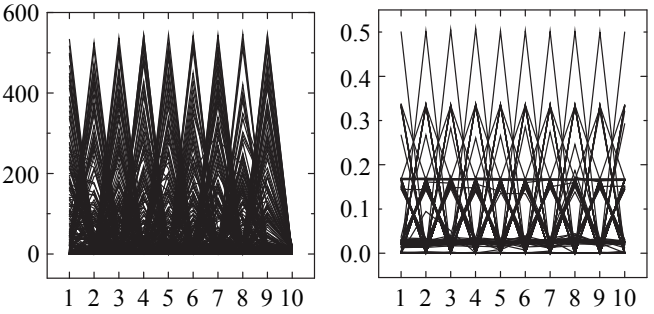
(a) NSGA-II. (b) NSGA-III.

Fig. 7. Obtained solution sets on the five-objective DTLZ1 problem.



(a) NSGA-II. (b) NSGA-III.

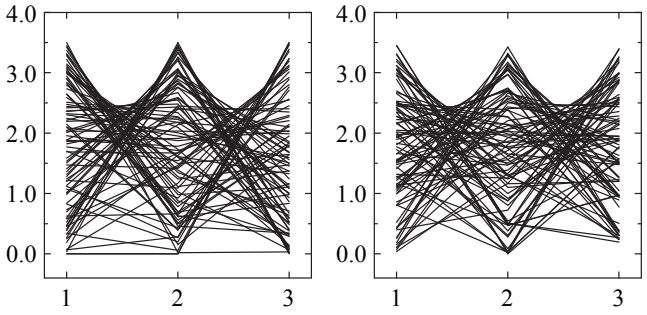
Fig. 8. Obtained solution sets on the eight-objective DTLZ1 problem.



(a) NSGA-II. (b) NSGA-III.

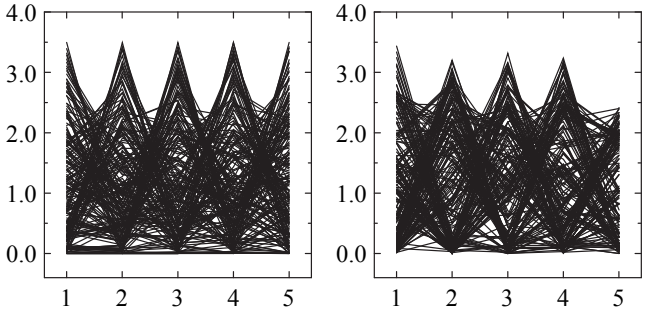
Fig. 9. Obtained solution sets on the ten-objective DTLZ1 problem.

Figs. 6-9 also demonstrate high diversification ability of NSGA-II. This ability plays an important role on its application to the Max-DTLZ1-4 problems. Experimental results on the Max-DTLZ2 problems are shown in Figs. 10-13.



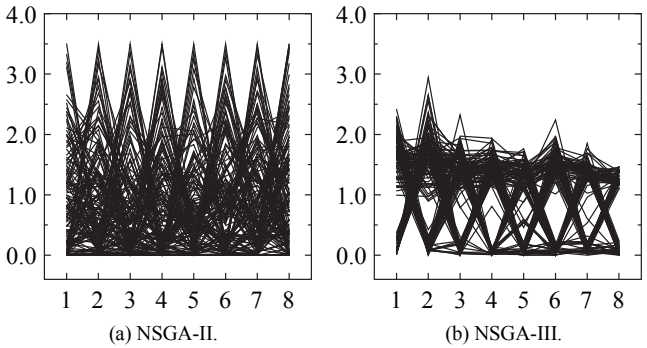
(a) NSGA-II. (b) NSGA-III.

Fig. 10. Obtained solution sets on the three-objective Max-DTLZ2 problem.



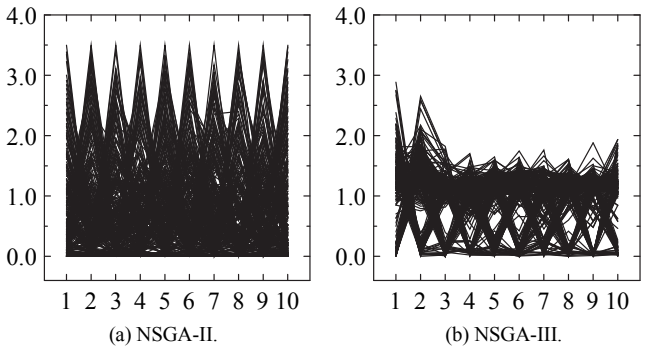
(a) NSGA-II. (b) NSGA-III.

Fig. 11. Obtained solution sets on the five-objective Max-DTLZ2 problem.



(a) NSGA-II. (b) NSGA-III.

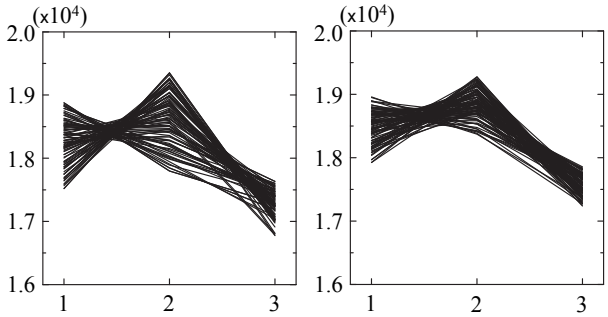
Fig. 12. Obtained solution sets on the eight-objective Max-DTLZ2 problem.



(a) NSGA-II. (b) NSGA-III.

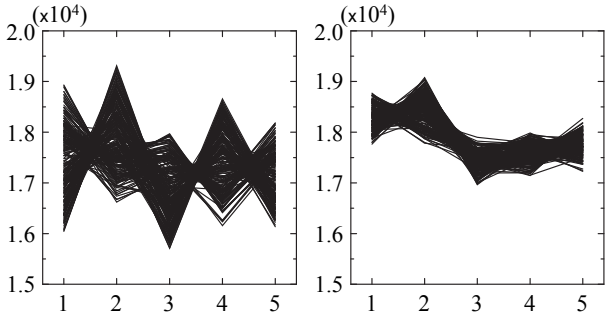
Fig. 13. Obtained solution sets on the ten-objective Max-DTLZ2 problem.

Figs. 14-17 show experimental results on the knapsack problems for the case of α = 0.2 (weak correlation among the objectives). Solution sets with larger diversity are obtained by NSGA-II in Figs. 14-17.



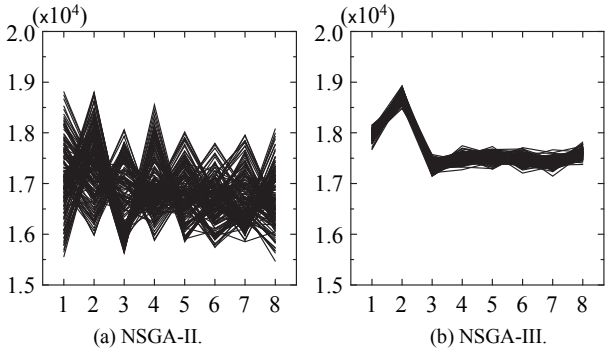
(a) NSGA-II. (b) NSGA-III.

Fig. 14. Obtained solution sets on the three-objective knapsack problem.



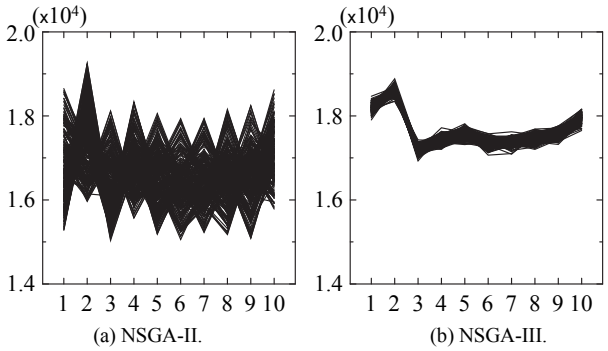
(a) NSGA-II. (b) NSGA-III.

Fig. 15. Obtained solution sets on the five-objective knapsack problem.



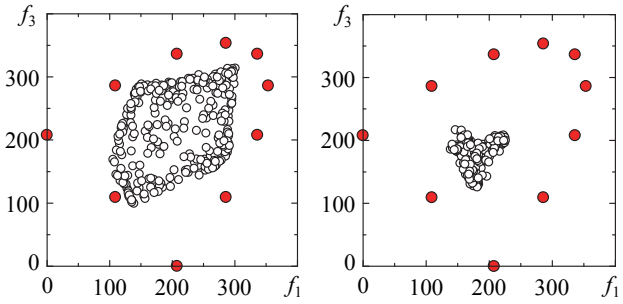
(a) NSGA-II. (b) NSGA-III.

Fig. 16. Obtained solution sets on the eight-objective knapsack problem.



(a) NSGA-II. (b) NSGA-III.

Fig. 17. Obtained solution sets on the ten-objective knapsack problem.



(a) NSGA-II. (b) NSGA-III.

Fig. 18. Solution sets on the ten-objective 100D distance minimization.

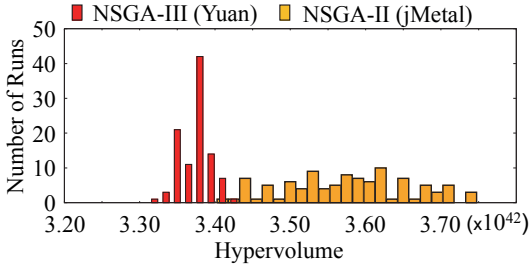


Fig. 19. Histogram of the obtained 100 hypervolume values.

Fig. 18 shows the obtained solution sets for the ten-objective 100D distance minimization problem. The solution set by NSGA-II has much larger diversity. However, the larger diversity does not always mean the larger hypervolume values. Actually, better results are obtained by NSGA-III for the distance minimization problems in Table V whereas better results are obtained by NSGA-II for the knapsack problems. For further comparing NSGA-II with NSGA-III, we apply each algorithm to the ten-objective knapsack problem with $\alpha = 0.2$ 100 times. Experimental results are summarized in Fig. 19.

V. Concluding Remarks

We demonstrated that NSGA-III did not always outperform NSGA-II when they were compared on various many-objective test problems. Whereas NSGA-III outperformed NSGA-II on the frequently-used DTLZ1-4 test problems, its superiority was unclear on the maximization DTLZ1-4 test problems. Better results were obtained by NSGA-II on many-objective knapsack problems. These observations suggest that the choice of test problems has a larger effect on performance comparison results than the number of objectives. Another important observation was that totally different experimental results were obtained from different implementations of NSGA-III (jMetal and Yuan [19]). Small differences are unavoidable since the source code of the original NSGA-III algorithm is not available. However, since their difference was large, we examined each code very carefully. Then, we found a possible reason for their difference, which was explained by a small difference in the formulations of the normalization between the NSGA-III paper [8] and its early access version. As shown in Fig. 5, similar results were obtained by using the same formulation for the normalization,

References

[1] K. Deb, S. Agrawal, A. Pratap, and T Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Proc. of 6th International Conference on Parallel Problem Solving from Nature - PPSN VI*, pp 849-858, Paris, France, September 18-20, 2000.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, 2001.

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

[4] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.

[5] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," *Proc. of 2008 IEEE Congress on Evolutionary Computation*, pp. 2419-2426, Hong Kong, China, June 1-6, 2008.

[6] C. von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707-756, July 2014.

[7] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 48, no. 1, Article 13, pp. 1-35, September 2015.

[8] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, Part I: Solving problems with box constraints," *IEEE Trans. on Evolutionary Computation*, vol. 18, no. 4, pp. 577-601, August 2014.

[9] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 19, no. 3, pp. 445-460, June 2015.

[10] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. on Evolutionary Computation*, vol. 19, no. 5, pp. 694-716, October 2015.

[11] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 20, no. 1, pp. 16-37, February 2016.

[12] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," *Proc. of 2002 IEEE Congress on Evolutionary Computation*, pp. 825-830, May 12-17, 2002.

[13] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 5, pp. 477-506, October 2006.

[14] H. Ishibuchi, H. Masuda, and Y. Nojima, "Pareto fronts of many-objective degenerate test problems," *IEEE Trans. on Evolutionary Computation* (available from IEEE Xplore as an early access paper).

[15] O. Schütze, A. Lara, and C. A. C. Coello, "On the influence of the number of objectives on the hardness of a multiobjective optimization problem," *IEEE Trans. on Evolutionary Computation*, vol. 15, no. 4, pp. 444-455, August 2011.

[16] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Trans. on Evolutionary Computation*, vol. 19, no. 2, pp. 264-283, April 2015.

[17] H. Masuda, Y. Nojima, and H. Ishibuchi, "Visual examination of the behavior of EMO algorithms for many-objective optimization with many decision variables," *Proc. of 2014 IEEE Congress on Evolutionary Computation*, pp. 2633-2640, Beijing, China, July 6-11, 2014.

[18] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760-771, October 2011. Codes were downloaded on January 15, 2016 from jMetal 5 Web Site (http://jmetal.github.io/jMetal/).

[19] http://learn.tsinghua.edu.cn:8080/2012310563/ManyEAs.rar (Codes were downloaded on October 1, 2015).

[20] H. Ishibuchi, N. Akedo, and Y. Nojima, "A many-objective test problem for visually examining diversity maintenance behavior in a decision space," *Proc. of 2011 Genetic and Evolutionary Computation Conference*, pp. 649-656, Dublin, Ireland, July 12-16, 2011.