Corner Sort for Pareto-Based Many-Objective Optimization

Handing Wang, *Student Member, IEEE,* Xin Yao, *Fellow, IEEE*

***Abstract*—Nondominated sorting plays an important role in Pareto-based multiobjective evolutionary algorithms (MOEAs). When faced with many-objective optimization problems multi-objective optimization problems (MOPs) with more than three objectives, the number of comparisons needed in nondominated sorting becomes very large. In view of this, a new corner sort is proposed in this paper. Corner sort first adopts a fast and simple method to obtain a nondominated solution from the corner solutions, and then uses the nondominated solution to ignore the solutions dominated by it to save comparisons. Obtaining the nondominated solutions requires much fewer objective comparisons in corner sort. In order to evaluate its performance, several state-of-the-art nondominated sorts are compared with our corner sort on three kinds of artificial solution sets of MOPs and the solution sets generated from MOEAs on benchmark problems. On one hand, the experiments on artificial solution sets show the performance on the solution sets with different distributions. On the other hand, the experiments on the solution sets generated from MOEAs show the influence that different sorts bring to MOEAs. The results show that corner sort performs well, especially on many-objective optimization problems. Corner sort uses fewer comparisons than others.**

***Index Terms*—Corner sort, many-objective optimization, nondominated sort, Pareto-based multiobjective evolutionary algorithms (MOEA).**

I. Introduction

NON-DOMINATED sorting is an essential component of Pareto-based multiobjective evolutionary algorithms (MOEAs), which plays an important role in the development of Pareto-based MOEAs. Pareto-based MOEA is a kind of MOEAs with the selection operator based on Pareto dominance, such as MOGA [1], NSGA [2], NSGA-II [3], NPGA [4], SPEA [5], SPEA2 [6], PAES [7], PESA-II [8], MOPSO [9], and TDEA [10]. As the development of Pareto-based MOEAs so far, nondominated sorting is still a key topic of Pareto-based MOEAs, because most computation cost of Pareto-based MOEAs comes from nondominated sorting.

Nondominated sorting is a process of assigning solutions to different ranks. According to Pareto dominance, the solutions in the same rank are nondominated by each other, and they are dominated by at least one solution in their former rank. Nondominated sorting is required to output the right Pareto dominance rank. All the nondominated sorts output the same result by inputting the same solution set. The only difference is their different computational cost. The time complexity of the native nondominated sorting is $O(mN^3)$, where m is the number of objectives and N is the magnitude of solution set. Deb [3] proposed a fast nondominated sort, which lowers the complexity to $O(mN^2)$. However, there is still some waste on some unnecessary comparisons.

Many-objective optimization problem is a kind of special MOP with more than three objectives [11]. Its large number of objectives increase its difficulties for Pareto-based MOEAs due to their weak Pareto dominance selection pressure [12]–[15]. Objective reduction [16], [17], and scalarization methods [18] are some strategies to reduce the difficulties of the original problems. Also, many MOEAs aim to solve many-objective optimization problems, such as IBEA [19], HypE [20], and MODELS [21]. Among the studies of many-objective optimization, dominance strategy and ranking are mainly focused on. For example, a controlling dominance area is proposed to strengthen the selection for many-objective optimization problems in [22], an epsilon-ranking is adopted to improve Pareto dominance in [23], a kind of fuzzy dominance is applied in [24] and [25], different kinds of data structure and representation increase the speed of ranking in [26] and [27].

In view of the characteristics of many-objective optimization problems, nondominated sort has to face new challenges because of the large number of objective comparisons. The challenges come from two aspects. First, as the number of objectives increases, more objective comparisons are needed to determine their dominance relation. Secondly, the existing comparison saving method cannot work effectively on many-objective optimization problems, because there are fewer dominated solutions to be ignored in the solution set. In view of this, a corner sort is proposed specially for the comparison saving of many-objective optimization problems. Thus, MOEAs can save more computational cost (both time and space) for solving many-objective optimization problems. The common comparison saving method is also adopted in corner sort. Corner sort saves the comparisons in the process of

Manuscript received August 14, 2012; revised February 4, 2013; accepted February 12, 2013. Date of publication March 7, 2013; date of current version December 12, 2013. This work was supported in part by an EU FP7 IRSES, under Grant 247619, on Nature Inspired Computation and its Applications (NICaiA), the National Natural Science Foundation of China, under Grant 61001202, and the National Research Foundation for the Doctoral Program of Higher Education of China under Grant 20100203120008. The work of X. Yao was supported by the Royal Society Wolfson Research Merit Award. This paper was recommended by Associate Editor H. Ishibuchi.

H. Wang is with Xidian University, Xi'an 710071, China (e-mail: wanghanding@163.com).

X. Yao is with the University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2013.2247594

obtaining the nondominated solutions. It discards the solutions dominated by the nondominated solutions in the sorting for the current rank. The contributions of this paper include the following.

1) Corner sort adopts a simple and fast way to obtain nondominated solutions according to a characteristic of MOPs (the solution of the best objective is nondominated). It only requires $N-1$ objective comparisons for the set of N solutions. The corner solutions are preferentially selected to obtain the nondominated solutions, which is the reason why the proposed sort is called corner sort.
2) In our experiments, the number of the comparisons between two objectives rather than the number of the dominance comparisons between two solutions is employed to evaluate the performance of the different nondominated sorts, which describes the computational cost fairly.

The rest of this paper is organized as follows. The related concepts have been introduced in Section II. The basic idea and details of corner sort are given in Section III. Section IV is the experimental part to show the performance of corner sort. Finally, Section V concludes the whole paper.

II. Background

A. Pareto Dominance

Pareto dominance is defined for comparing two solutions with multiple objectives, which is the base of nondominated sorting. One vector is said to dominate another one only if all of its objectives are not worse than another ones. Vectors $x = (x_1, ..., x_i, ..., x_m)$ and $y = (y_1, ..., y_i, ..., y_m)$ are two solutions of a minimization problem with m objectives. If $x_i \leq y_i (1 \leq i \leq m)$ and $x \neq y$, x dominates y, written as $x \prec y$.

Usually, dominance comparison is applied by comparing the objective values of two solutions sequentially. Theoretically, it needs m objective comparisons to determine the relation between two solutions for an MOP with m objectives. In fact, an effective method is applied in all the existing nondominated sort. As the objective comparisons are serial, the dominance relation is regarded as nondominated once meeting conflict. Thus, it needs 2 to m objective comparisons. In other words, the cost of different solutions is different. It is unfair to employ the number of comparisons of two solutions to evaluate the performance of nondominated sort. In the experiment of Section IV, we employ the number of objective comparisons to evaluate the performance of different nondominated sorts.

B. Corner Solutions

For an MOP with m objectives, the solution that is considered k objectives ($k < m$) in the m-objective space is called a corner solution [28]. In this paper, we only consider the corner solutions with $k = 1$. There are some examples of corner solutions in Fig. 1.

C. Existing Nondominated Sort Algorithms

Nondominated sort is one of the most complicated processes in Pareto-based MOEAs. Its natural time complexity is $O(mN^3)$, which is very high for the solution set of a large scale. All nondominated sorts aim at obtaining the right dominance rank of the solution set at the same time of lowering the number of comparisons.

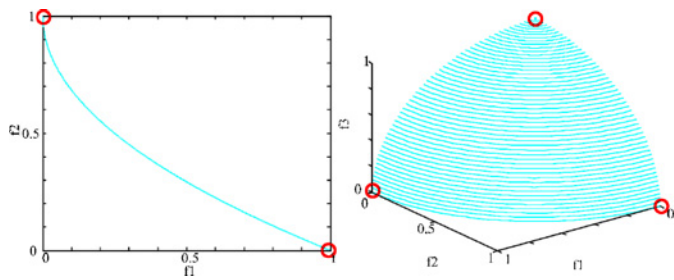


Fig. 1. Examples of corner solutions.

Fast nondominated sort [3] is the first fast algorithm to lower the complexity to $O(mN^2)$. As [29] shows, no-free-lunch does not hold all the time. We can still find generally outperforming sorting algorithms. There are still some unnecessary comparisons in fast nondominated sort. In the later study, researchers devoted themselves to reducing the unnecessary comparisons in the sorting. Among their research, nondominated rank sort [30] and deductive sort [31] are the examples of $O(mN^2)$ nondominated sorts with some comparison saving strategies.

1) *Fast Nondominated Sort:* Fast nondominated sort [3] is the promotion of the development of Pareto-based MOEAs. This first travels all of the solutions to obtain all the dominance relation between every two solutions. According to such a relation, the dominance rank is finally assigned. Fast nondominated sort needs at most $mN(N-1)$ objective comparisons. For every solution p in fast nondominated sort, list S_p stores the solutions that are dominated by solution p, and counter n_p records the number of the solutions that dominate solution p. The space complexity is $O(N^2)$. When the scale of solution set increases, both time and space complexity increase rapidly.

2) *Nondominated Rank Sort:* Nondominated rank sort in [30] is a sort with a comparison saving strategy, which is faster than fast nondominated sort. It sorts the nondominated solutions in the current solution set sequentially. If one solution is dominated by any solution in the current solution set, its following comparisons to the rest solutions are ignored. If one solution is nondominated in the current solution set, it is marked as the current rank and deleted in the current solution set. In the worst case, nondominated rank sort requires $mN(N-1)/2$ objective comparisons. The time complexity of nondominated rank sort is still $O(mN^2)$. It does ignore some unnecessary comparisons for the dominated solutions, which requires less cost than fast nondominated sort. Furthermore, its space complexity is only $O(N)$.

3) *Deductive Sort:* Although nondominated rank sort requires fewer comparisons, it still wastes some comparisons. For example, if $a \prec b, b \prec c$, then $a \prec c$. The comparisons between a and c are unnecessary. Deductive sort [31] aims at saving these comparisons. On one hand, it ignores the comparisons of the dominated solutions to the rest solutions as nondominated rank sort. On the other hand, it marked and ignored any dominated solutions in the ranking of current rank.

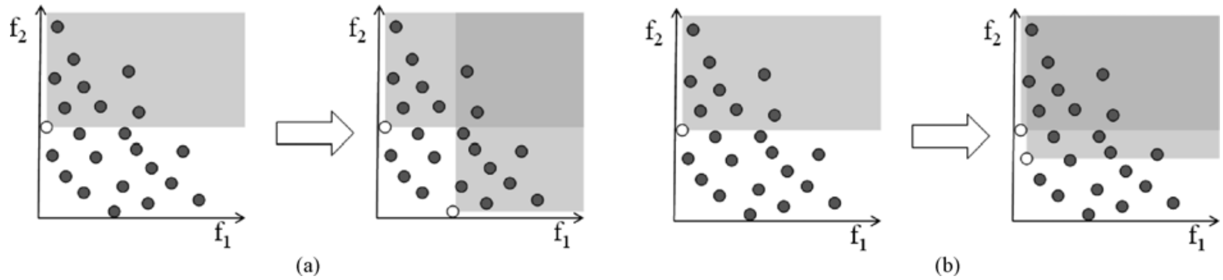


Fig. 2. Two cases of choosing nondominated solution.

Similar to nondominated rank sort, deductive sort requires $mN(N-1)/2$ objective comparisons in the worst case. Its time complexity is $O(mN^2)$ and its space complexity is $O(N)$.

III. Corner Sort

A. Basic Idea

The basic idea of saving comparisons in corner sort is to use the nondominated solutions to ignore the solutions that they dominate. Two processes of corner sort combine together for the aim of comparison saving. One ignores the dominated solutions as that in deductive sort. The other one obtains the nondominated solutions, which is unique to corner sort. Naturally, it needs $m(N-1)$ at most and $2(N-1)$ at least objective comparisons for obtaining a nondominated solution in the solution set of N solutions. We wonder if some characteristics of MOP can help to save comparisons in this process. We find that the solutions with the best objective are definitely nondominated. In other words, corner solutions [28] are selected preferentially for nondominated solutions.

Because only $N-1$ comparisons on only one objective are needed for obtaining a nondominated solution, it is much fewer than the $m(N-1)$ objective comparisons especially for many-objective optimization problems. Although it saves comparisons, its time complexity is still $O(mN^2)$. Its space complexity is $O(N)$.

After obtaining one nondominated solution, corner sort marks the solutions dominated by the obtained nondominated solution to ignore in the later sort of the current rank. As the nondominated solutions have been ranked, their comparisons to the unmarked solutions are only to check whether they dominate the unmarked solutions.

B. Corner Sort

The implementation of corner sort is a loop of two steps, obtaining a nondominated solution and marking the solutions that are dominated by the nondominated solution. For an MOP with m objectives, at least m nondominated solutions are available. For a sequential process, only one nondominated solution is needed for each time. It is worth discussing the order of choosing nondominated solutions. The best order depends on the distribution of the solution set. However, the situation cannot be estimated in advance. The uniform distribution is the average of all the situations. Therefore, we use the solution set with a kind of uniform distribution for all the situations to analyze this problem. There are two cases as shown in Fig. 2, where the white points are nondominated solutions and the light grey area is the area that they dominate. Case A is to choose the nondominated solutions in a looping order and Case B is to choose the nondominated solutions always in one objective. In Fig. 2, it is clear that the marked area in Case A is larger than that of Case B. The looping order is adopted for choosing the nondominated solutions in corner sort. It does not mean Case B is useless. In some particular situations that there are more solutions around objective f_1, Case B performs better than Case A. However, if there are more solutions around other objectives, Case B performs least-effectively. Taken all these different kinds of distribution into consideration, it is impossible to require that our sort has the best efficiency in all different situations. Case A is not the best order for all the situations but the most robust order, because it never acts too badly on these solution sets with any particular distribution.

The pseudocode of corner sort is given in Table I. In Table I, if the nondominated solution is obtained from objective f_j, objective f_j is ignored in its later comparisons to other unmarked solutions. As the nondominated solution has the best objective f_j, the comparisons on objective f_j is unnecessary. In order to explain the process of corner sort, an sorting example is included in Fig. 3. First, all the solutions in P are unmarked, corner sort obtains the nondominated solution and marks the solutions that the nondominated solution dominates. It chooses the nondominated solutions in a looping order as in Fig. 3(A). It finishes sorting of rank one when all the solutions are all marked as in Fig. 3(B), the white points are in rank one. After that all the unranked solutions are unmarked for the sorting of the next rank as in Fig. 3(C). Corner sort loopingly acts as above until all the solutions are ranked.

C. Corner Sort for Many-Objective Optimization Problems

For many-objective optimization problems, the cost of nondominated sorting increases rapidly with the number of objectives. The comparisons between two solutions of many-objective optimization problems are very expensive, because they have more objectives to compare. In many-objective optimization problems, the solutions are difficult to dominate. Thus, there is less chance to save comparisons by ignoring the dominated solutions. Corner sort aims to save comparisons from the process of obtaining the nondominated solutions. It only requires $N-1$ comparisons on one objective. Normally, 2 to m objective comparisons are needed for the comparison

TABLE I
PSEUDOCODE OF CORNER SORT

```
Corner sort
Parameters: P-Solution set, N-The number of solutions, Rank-Rank result, m-Number of objectives.
Rank[1 : N] = 0
i = 1
Do
    Unmark all the unranked solutions(whose ranks are 0), j = 1
    Do
        Find solution P[q] of the best objective f among the unmarked ones
        mark q, Rank[q] = i
        j = (j + 1)%m + 1 // Loop objectives
        For k = 1 : N
            If P[k] is unmarked and P[q] ≺ P[k]
                Mark P[k]
            End
        End
    Until all the solutions in P are marked
    i = i + 1
Until all the solutions in P are ranked
```

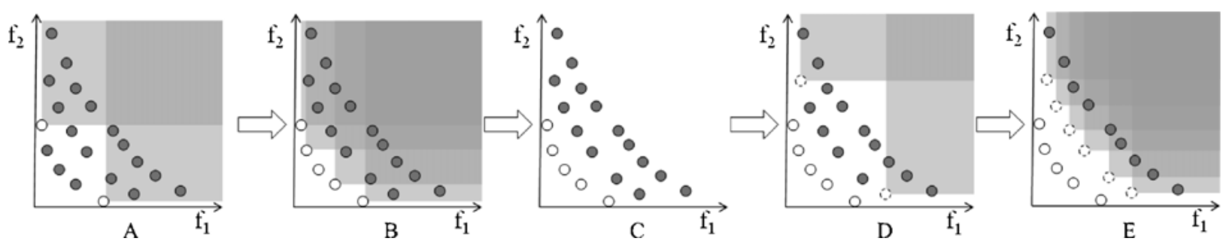


Fig. 3. Example of corner sort. (A) Corner solutions are ranked and the solutions dominated by them are ignored. (B) Ranking for the first rank is finished. (C) Unranked solutions are unmarked. (D) Corner solutions of the rest unmarked solutions are ranked and the solutions dominated by them are ignored. (E) Ranking for the second rank is finished.

of two solutions. $N-1$ comparisons on one single objective are definitely fewer than $m(N-1)$ objective comparisons. The more objectives one MOP has, the more objective comparisons corner sort can save.

In the real world, there are many-objective optimization problems based on preference. Corner sort is suitable for the preference based many-objective optimization. The order of choosing nondominated solution can be referred to preference information (such as weight vector and lexicographic minimum). In this way, the sorting is more efficient.

IV. Experimental Studies

We test the performance of different nondominated sorts on both artificial datasets and the solution sets from MOEAs. On one hand, the artificial datasets describe the situation that MOEAs may meet. As only the objective values are considered in nondominated sorts, the artificial datasets only include objective values. The experiments on those datasets aim to analyze the behaviors of different sorts on the solution sets with particular distribution. On the other hand, the experiments on the solution sets from MOEAs aim to show the influence that different sorts bring to MOEAs. Fast nondominated sort [3], nondominated rank sort [30], and deductive sort [31] are employed as the compared algorithms in our experiment. As the process of obtaining nondominated solutions does not need to compare all the objectives of two solutions, it is not available

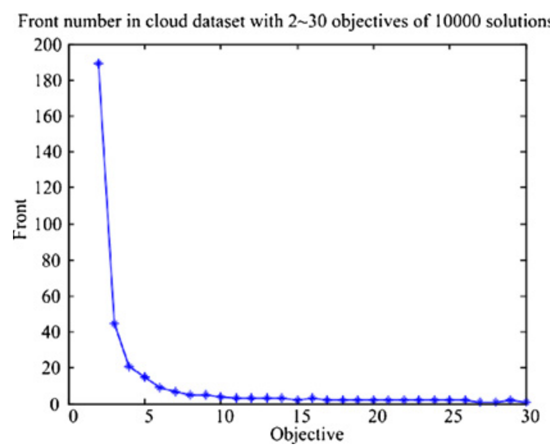


Fig. 4. Number of fronts in the cloud dataset with 2–30 objectives of 10 000 solutions.

to adopt the number of comparisons between two solutions to evaluate the performance. We use the number of objective comparisons and execution time to evaluate the performance of different nondominated sorts. It is worth noting that the objective comparisons for obtaining nondominated solutions in corner sort are included in the total number of objective comparisons. Since all the nondominated sorts obtain the same Pareto dominance rank for the same solution set and the aim of our experiments is to evaluate the efficiency of different sorts, the results of ranking are not included in this paper.

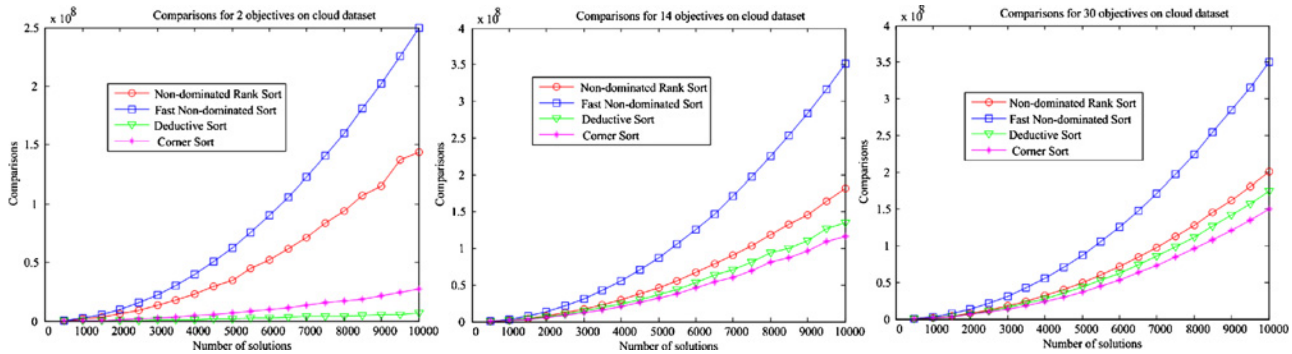


Fig. 5. Comparisons of 4 nondominated sort on 2, 14, 30 objective cloud datasets with different numbers of solutions.

TABLE II
SAMPLE CREATION ALGORITHM FOR CLOUD DATASET

Sample creation algorithm for cloud dataset
Parameters: N—The number of solutions, m—The number of objectives, U(0, 1)—Uniformly distributed number in [0, 1]
For i=1:N
For j=1: m
Dataset(i, j)= U(0, 1)
End
End

A. *Experiments on Cloud Dataset*

1) *Cloud Dataset:* Cloud dataset is a dataset of uniform-random distributed solutions, which describes the random situation in Pareto-based MOEAs. The sample creation algorithm is shown in Table II. Such a case usually occurs at the beginning of MOEAs. The relation between the number of fronts and the number of objectives is shown in Fig. 4. With the increasing of objectives, the number of fronts decreases rapidly, because most solutions are nondominated by each other in many-objective optimization problems.

2) *Results:* The computational complexity of nondominated sort comes from two aspects, numbers of solutions and objectives. The results are divided into two parts, one is the experiment on the dataset of a fixed number of objectives with variable numbers of solutions and the other one is the experiment on the dataset of a fixed number of solutions with variable numbers of objectives.

Fig. 5 is the result of numbers of objective comparisons with variable numbers of solutions. According to Fig. 5, fast nondominated sort requires the most objective comparisons. Nondominated rank sort requires fewer objective comparisons than fast nondominated sort, but more than deductive sort and corner sort. For low-dimensional objectives such as two objectives, deductive sort requires fewer comparisons than corner sort. For high-dimensional objectives, deductive sort requires more comparisons than corner sort. Also, the performance of deductive sort and corner sort on the cloud dataset of different numbers of objectives is different by Fig. 5. This is because the dominance relation in the cloud dataset with different numbers of objectives is different. Since there are more fronts in the 2-objective cloud dataset, deductive sort can ignore more dominated solutions. Because of its ignoring strategy, deductive sort ignores solutions in a random way. Deductive sort requires fewer objective comparisons than corner sort in the low-dimensional cases.

Fig. 6 shows the average execution time of 4 nondominated sorts on cloud dataset with different numbers of solutions from 30 independent runs. Fast nondominated sort costs the most execution time, which is the same as the result of numbers of comparisons. Deductive sort costs more time than nondominated rank sort. Corner sort costs the least time in most cases.

According to Figs. 5 and 6, all the sorts require more computational cost with the increasing numbers of solutions. Fast nondominated sort increases most rapidly. Nondominated rank sort increases more slowly than fast nondominated sort, because it discards the nondominated ones in the later sorting. As both deductive sort and corner sort ignore the dominated solutions in the sorting of one rank, they use fewer objective comparisons.

Fig. 7(a) shows the result of numbers of objective comparisons with variable numbers of objectives. The number of objective comparisons of all the nondominated sorts increases with the objective number. For the datasets with more than 15 objectives, the objective comparison stops increasing, because cloud datasets with more than 15 objectives have only one front, which means there are no dominated solutions to ignore. It is worth noting that the number of objective comparisons of deductive sort increases faster than that of corner sort. Although corner sort requires more comparisons than deductive sort for MOPs with low-dimensional objectives, it requires fewer comparisons for MOPs with high-dimensional objectives. Fig. 7(b) shows the average execution time with variable numbers of objectives from 30 independent runs. It costs a little bit more execution time for the dataset with more objectives. Fast nondominated sort costs the most execution time, and corner sort costs the least execution time.

B. *Experiments on Fixed Front Dataset*

1) *Fixed Front Dataset:* Fixed front dataset [31] is a kind of dataset with a controllable number of fronts, which describes the situation in Pareto-based MOEAs when the search mainly focuses on fronts. In the fixed front dataset, solutions are divided into a fixed number of fronts with almost the same size. Every front is distributed on a line or a plane as in Fig. 8. The details of the sample creation algorithm are shown in

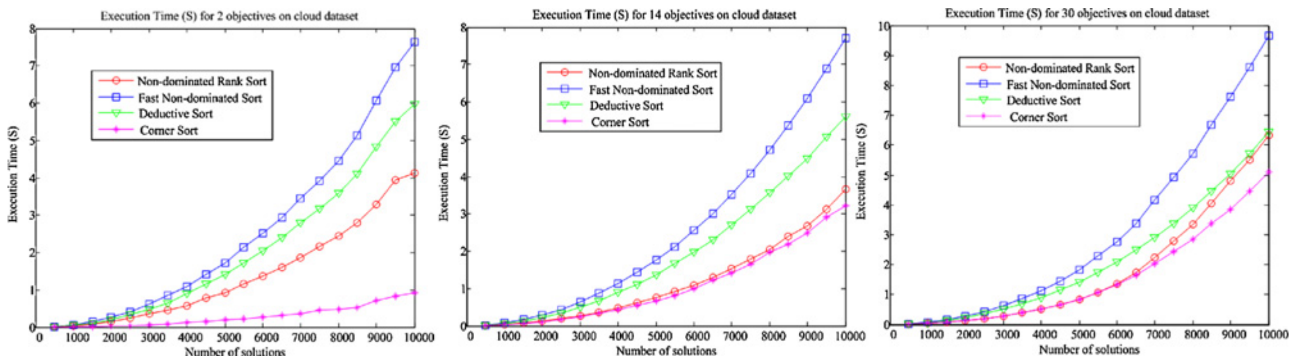


Fig. 6. Average execution time of 4 nondominated sort on 2, 14, 30 objective cloud datasets with different numbers of solutions from 30 independent runs.

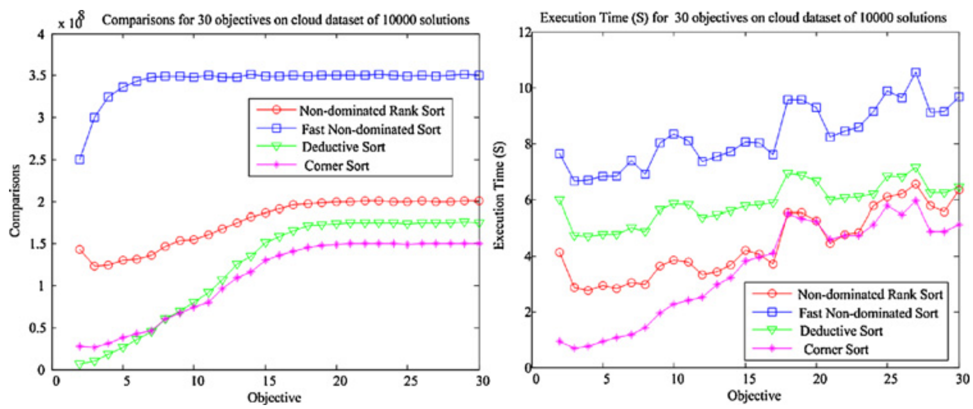


Fig. 7. Results of 4 nondominated sort on 2–30 objective cloud datasets of 10 000 solutions.

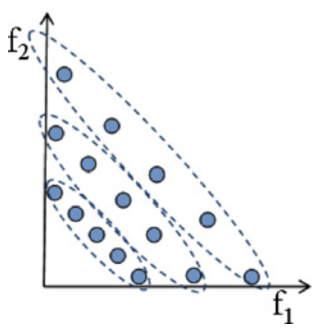


Fig. 8. Example of fixed front dataset is with three fronts and 15 solutions. The solutions in the area of dotted lines are in the same front.

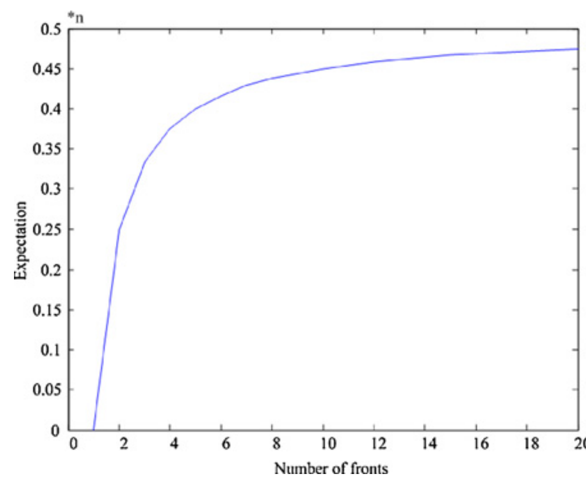


Fig. 9. Expected number of solutions that can be dominated with variable numbers of fronts in fixed front dataset.

Table III. In order to understand this dataset deeply, we take an example with F fronts and n solutions. For the solution in rank R, there are $n(F-R)/F$ solutions that are dominated by that solution. The solutions are distributed in different fronts with the same probability. Thus, the expectation of the solutions can be dominated as $\sum_{R=1}^{F} \frac{n}{F}\frac{F-R}{F} = n(\frac{1}{2} - \frac{1}{2F})$, which is shown in Fig. 9.

2) *Results:* In the experiment on the fixed front dataset, two factors influence the behavior of nondominated sort very much. One is the number of objectives, the other is the number of fronts. The following experiment has two parts for those two factors. As the influence of the number of solutions is analyzed in the experiment of cloud dataset. The fixed front datasets employed here are of a fixed number of solutions (7500).

Fig. 10 is the result of numbers of comparisons on the fixed front dataset with variable numbers of fronts. The numbers of comparisons of all the sorts drop with the increasing number of fronts except for fast nondominated sort, because there are more solutions that can be dominated in the datasets with larger number of fronts as Fig. 9. It is clear that fast nondominated sort requires the most comparisons in all cases of the experiment. Deductive sort and corner sort require fewer comparisons than nondominated rank sort. In most cases, corner sort requires fewer comparisons than deductive sort, except for the cases with low-dimensional objectives. It needs 2 to m objective comparisons between two solutions. For the high-dimensional cases in Fig. 10, we find that the

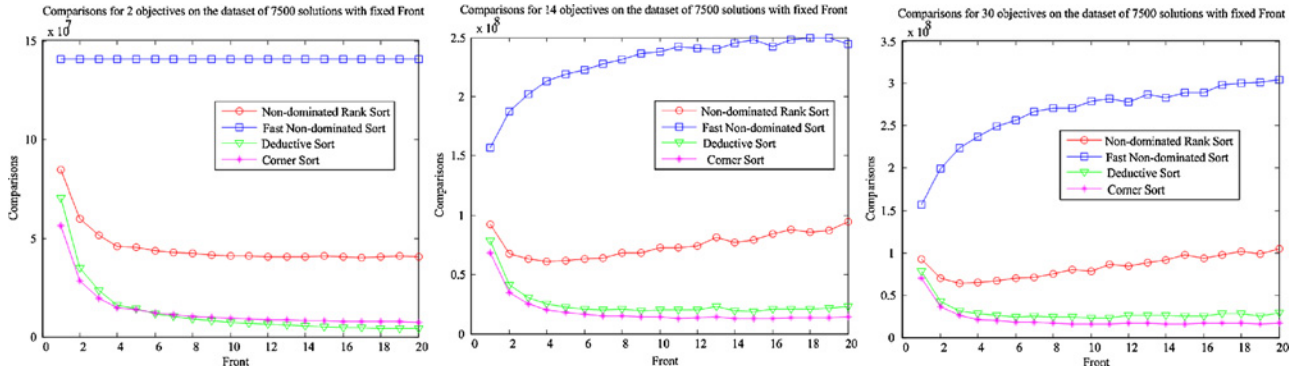


Fig. 10. Comparisons of 4 nondominated sort on the fixed front datasets with 1∼20 fronts of 7500 solutions.

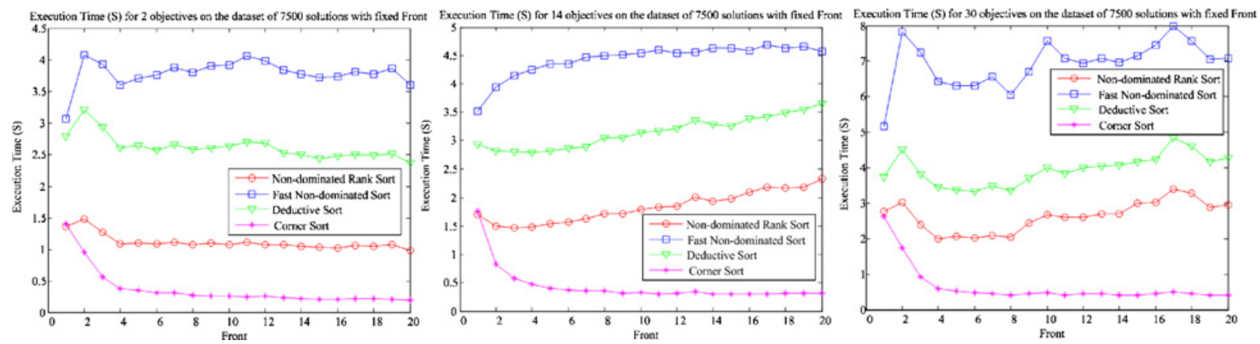


Fig. 11. Average execution time of 4 nondominated sort on the fixed front datasets with 1 ∼20 fronts of 7500 solutions from 30 independent runs.

number of objective comparisons of fast nondominated sort and nondominated rank sort increase a little bit with the number of fronts after their decreasing, because they do not ignore the dominated solutions. Fast nondominated sort ignores no solutions. Nondominated rank sort just ignores a part of comparisons of the dominated solutions.

Fig. 11 shows the result of average execution time on the fixed front dataset with variable numbers of fronts from 30 independent runs. Generally, the number of fronts does not influence execution time very much. Similar to the result on the cloud dataset, fast nondominated sort costs the most execution time, deductive sort costs the second most execution time, and corner sort costs the least execution time among the four sorts.

Fig. 12 is the result of numbers of comparisons on the fixed front dataset with variable numbers of objectives. From the result, we find that only the number of comparisons of fast nondominated sort increases with the increasing number of objectives. Among these four kinds of sorts, corner sort requires the fewest comparisons.

Fig. 13 is the result of average execution time on the fixed front dataset with variable numbers of objectives from 30 independent runs.The execution time of four nondominated sorts is hardly influenced by the number of objectives. Corner sort costs the least time in most cases of the experiment.

C. Experiment on Mix Dataset

1) *Mix Dataset:* The last two kinds of datasets are uniformly distributed. Cloud dataset is distributed in the objective space. Fixed front dataset is distributed on fronts. In the practical process of MOEAs, solutions are not distributed so uniformly. In view of this, we add an experiment on nonuniform datasets, i.e., mix dataset to test whether the distribution influences the performance of corner sort. Mix dataset is built by adding small size of cloud dataset to fixed front dataset with one front as Fig. 14.

2) *Results:* Fig. 15(a) is the result of comparisons on the mix dataset with variable numbers of objectives. Fast

TABLE III

SAMPLE CREATION ALGORITHM FOR FIXED FRONT DATASET

Sample creation algorithm for cloud dataset
Parameters: N—Number of solutions, m—Number of objectives, f-number of fronts
U(0, 1)—Uniformly distributed number in [0, 1]
$N_1=\lceil N/f \rceil$ %the number of solutions on every front
$N_2 = mod(N, f)$ %the number of solutions on the last front
For i=1:N_1 %the first front
Dataset(i,1)=U(0, 1); For j=2:m-1
Dataset (i, j)=U(0, 1) *sum(Dataset(i,1:j-1))
End
Dataset(i,m)=1-sum(Dataset(i,1:m-1))
End
For i=2:$f-1$ %the 2nd-f-1-th front
For j=1:N_1
Dataset($j + N_1 * (i-1)$,:)=Dataset(j,:)*i
End
End
For i=1:N_2
Dataset($i + N_1 * (f-1)$,:)=Dataset(i,:)*f
End
Rearrange the order of the dataset randomly.

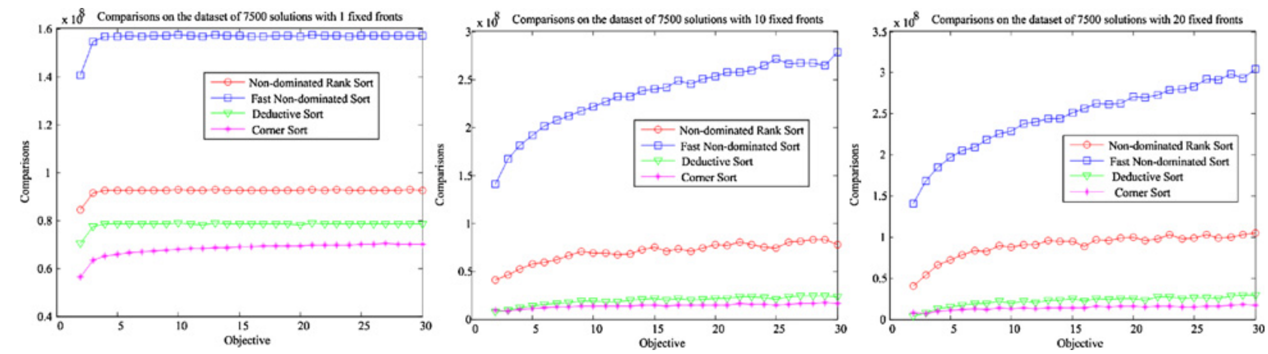


Fig. 12. Comparisons of four nondominated sort on 1, 10, and 20 fixed front datasets with 2–30 objectives of 7500 solutions.

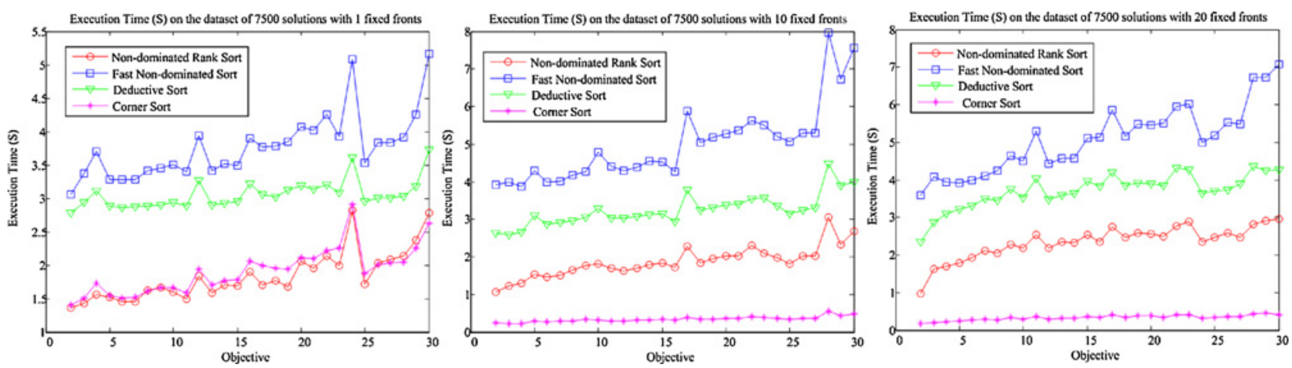


Fig. 13. Average execution time of four nondominated sort on 1, 10, and 20 fixed front datasets from 30 independent runs.

TABLE IV
PARAMETER SETTINGS FOR NSGA-II

Crossover	Mutation	Crossover ratio	Mutation ratio
SBX	Polynomial mutation	1	1/(dimension of decision variables)

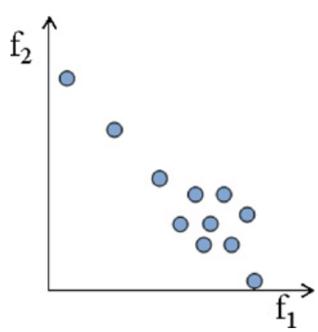


Fig. 14. Example of mix front dataset. Mix front is structured by a fixed front dataset (one front) and a cloud dataset of smaller size.

nondominated sort and nondominated ranking sort require the first and second most comparisons, respectively. For the mix dataset with low-dimensional objectives, corner sort requires more comparisons than deductive sort but the mix dataset with high-dimensional objectives, corner sort requires fewer comparisons than deductive sort, because it can save more objective comparisons in the process of obtaining nondominated solutions than deductive sort. Fig. 15(b) shows the result of execution time on the mix dataset with variable numbers of objectives. Fast nondominated sort costs the most time. Nondominated rank sort and corner sort cost the least time. In summary, uniformity does not influence nondominated sorts.

D. Experiment on Data Generated from MOEAs

1) *Data Generated from MOEAs:* All the above experiments are applied on the artificial datasets, which cannot reflect the performance of nondominated sorts in MOEAs. We adopted DTLZ1 ($|x_g| = 5$) and DTLZ2 ($|x_g| = 10$) with 2–30 objectives [32] for the experiment in this subsection. The data is from the populations of every generation in MOEAs. We run 200 generations on those benchmark problems with a population with 200 individuals in MOEAs. We adopt NSGA-II as the MOEA in the experiment. The parameter settings are shown in Table IV.

2) *Results:* There are 200 datasets for each problems. We conclude the results in Fig. 16 with a error bar plot, which describes statistical information. In Fig. 16, the numbers of comparisons of all the sorts except deductive sort have no influence with the number of objectives. Moreover, the execute time increases a little bit with the number of objectives. For DTLZ1, fast nondominated sort requires both the most comparisons and the longest time. Corner sort requires both the fewest comparisons and the shortest time. The result of DTLZ2 is similar to that of DTLZ1.

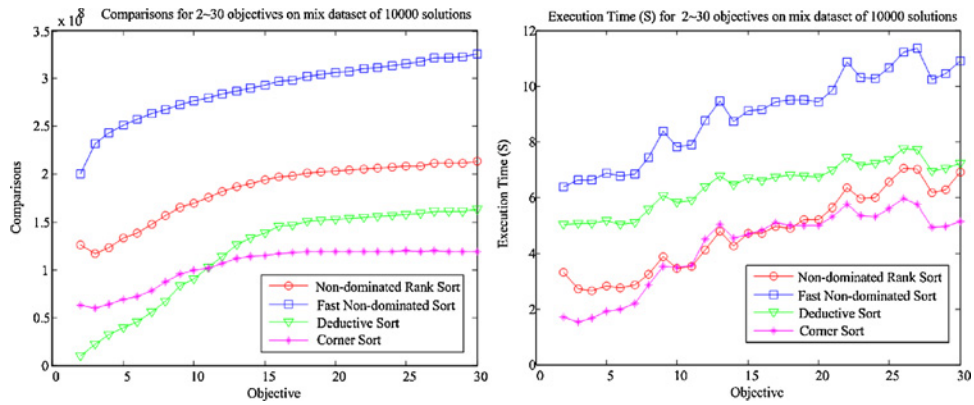


Fig. 15. Results of 4 nondominated sort on mix datasets with 2–30 objectives.

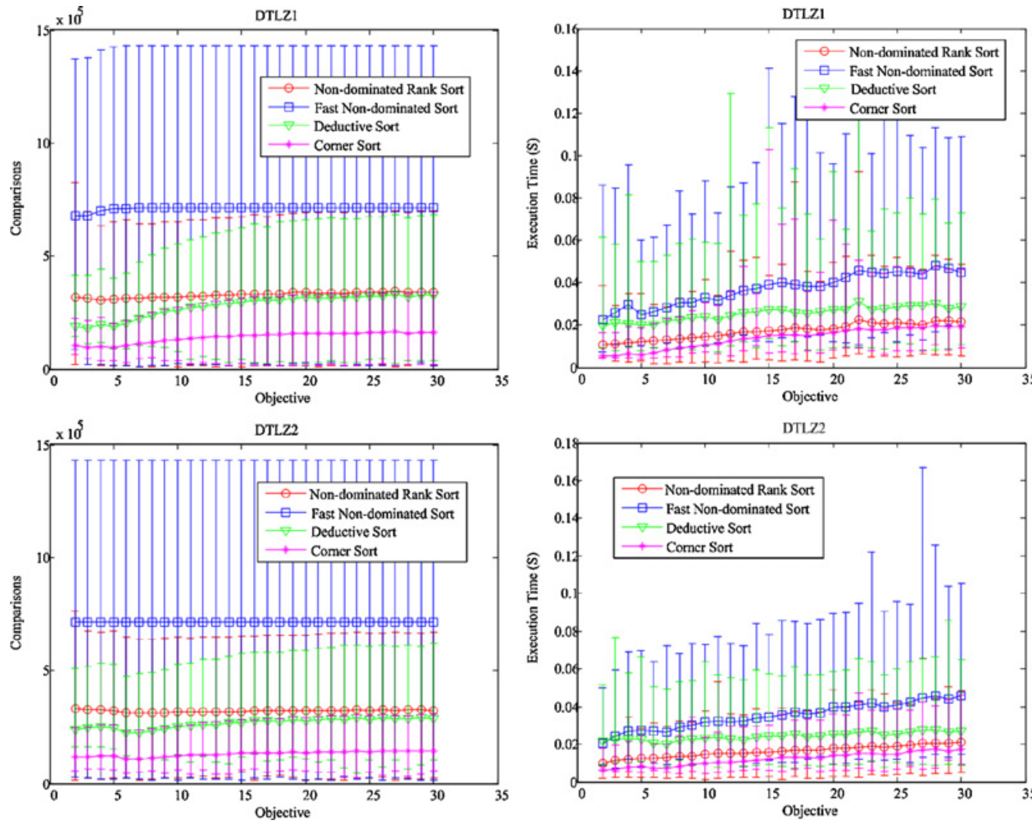


Fig. 16. Results of 4 nondominated sort on the datasets from MOEAs with 2–30 objectives.

Furthermore, the behaviors in different stages of MOEAs of different sorts are shown in Fig. 17. In Fig. 17, the number of comparisons is shown with different generations. Generally, corner sort performs best among these four sorts in all the stages. For 2-objective problems, nondominated rank sort has worse performance at the beginning than that in other stages. Corner sort and deductive sort has better performance at the beginning than that in other stages. For MOPs with high dimensional objectives, the stages have little influence on all the sorts.

E. Discussion and Analysis

From the above experiments, we find corner sort save more execution time and comparisons than other comparative sorts on many-objective optimization problems. The advantage of corner sort is in the process of obtaining nondominated solutions of many-objective problems. It only takes $N - 1$ objective comparisons, which is fewer than other sorts. In the beginning of MOEAs, which is simulated by cloud dataset, corner sort outperforms others on many-objective problems. In the end of MOEAs, which is simulated by fix front dataset, corner sort still saves the most time and comparisons. The distribution of dataset has no influence on the efficiency of corner sort, which is shown by the experiment on mixed dataset. From these experiments on artificial datasets, we conclude those above. The experiment on the solution set generated from MOEAs shows that corner sort saves the most computational cost in whole procession of MOEAs. Although the experiment on the benchmark problem reflects the behaviors of different sorts, the problems in the real

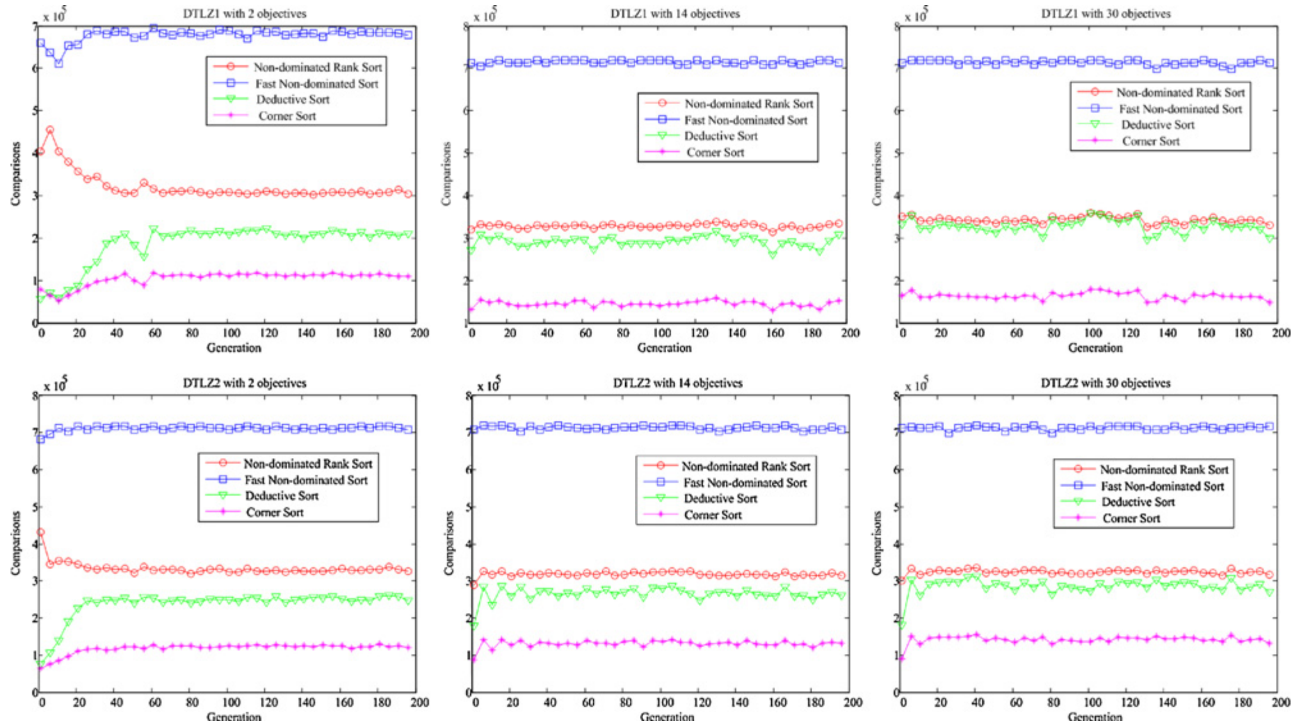


Fig. 17. Comparisons of 4 nondominated sort on 2, 14, 30 objectives using the datasets from MOEAs in different generations.

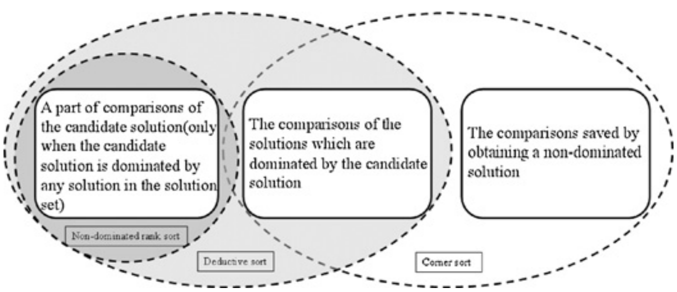


Fig. 18. Illustration of comparisons saved by different nondominated sorts.

world are different from the benchmark. However, we still can infer that unnormalized objectives would not influence the behavior of corner sort because of its separate sorting on objectives.

Theoretically, all those four comparative sorts have the same level of computational complexity in the worst case. In the worst case, fast dominated sort needs $mN(N-1)$ objective comparisons, while others need $mN(N-1)/2$ objective comparisons. Obviously, fast dominated sort is the worst one without any comparison saving strategies. Thus, the analysis for other sorts is necessary. Although they require the same number of objective comparisons in the worst case, they have different comparison saving methods. Fig. 18 shows their differences. All those sorts use a candidate solution to ignore the comparisons of the dominated solutions. However, the saved comparisons come from different ways. Nondominated rank sort only saves the comparisons to the rest solutions when the candidate solution is dominated by any solutions. Deductive sort also saves those comparisons. At the same time, it saves the comparisons of the solutions that are dominated by the candidate solution. Deductive sort saves more comparisons than nondominated sort definitely. Corner sort obtains a nondominated solution by only $N-1$ objective comparisons, which saves comparisons. As the candidate solution is nondominated, the first part of saved comparisons in Fig. 18 cannot be obtained by corner sort. By the nondominated candidate solution, corner sort saves the second part of comparisons in Fig. 18 as deductive sort. It seems to be difficult to compare deductive sort and corner sort, which depends on the situation of the solution set. Namely, when the third part saves more comparisons than the first one, corner sort is better than deductive sort. As the analysis of Section III-C shows, the more objectives an MOP has, the more objective comparisons corner sort saves by obtaining nondominated solutions. Moreover, there is less chance of a solution to be dominated in the solution set of many-objective problem. Few comparisons can be saved by the first part when faced many-objective problems. Therefore, corner sort saves more comparisons than deductive sort. This is the reason why corner sort performs best among the comparative sorts on many-objective optimization problems.

V. Conclusion

A novel corner sort for Pareto-based many-objective optimization was proposed in this paper. The proposed sort aimed at saving comparison when obtaining nondominated solutions. It saved more objective comparisons and execution time than other sorts on many-objective optimization problems according to our comparative experiments.

Although corner sort performed well in many-objective problems, its performances on MOPs with low-dimensional objectives were not satisfactory. We hope to improve corner sort further in our future work. As the proposed sort can obtain m nondominated solutions at the same time, we hope to develop the current sort into a parallel version. Moreover, the order of choosing nondominated solution in corner sort affects its behavior according to different cases of solution set. In the future, we hope to improve it by adding a self-adaptive one.

REFERENCES

[1] C. Fonseca and P. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algor.*, vol. 1. 1993, pp. 416–423.

[2] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolut. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

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

[4] J. Horn, N. Nafpliotis, and D. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. IEEE CEC*, 1994, pp. 82–87.

[5] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evolut. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[6] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Proc. EUROGEN Evolut. Methods Des., Optim. Control Appl. Industrial Problems*, 2001, pp. 1–21.

[7] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation," in *Proc. IEEE CEC*, vol. 1. 1999, pp. 98–105.

[8] D. Corne, N. Jerram, J. Knowles, and M. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. GECCO*, 2001, pp. 283–290.

[9] C. Coello Coello and M. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. IEEE CEC*, vol. 2. 2002, pp. 1051–1056.

[10] I. Karahan and M. Koksalan, "A territory defining multiobjective evolutionary algorithms and preference incorporation," *IEEE Trans. Evolut. Comput.*, vol. 14, no. 4, pp. 636–664, Aug. 2010.

[11] V. Khare, X. Yao, and K. Deb, "Performance scaling of multiobjective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization* (Lecture Notes in Computer Science Series), C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb (Eds.) Berlin/Heidelberg, Germany: Springer, 2003, vol. 2632, pp. 376–390.

[12] K. Praditwong and X. Yao, "How well do multiobjective evolutionary algorithms scale to large problems," in *Proc. IEEE CEC*, 2007, pp. 3959–3966.

[13] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2007, pp. 742–756.

[14] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proc. IEEE*, Jun. 2008, pp. 2419–2426.

[15] O. Schutze, A. Lara, and C. Coello, "On the influence of the number of objectives on the hardness of a multiobjective optimization problem," *IEEE Trans. Evolut. Comput.*, vol. 15, no. 4, pp. 444–455, Aug. 2011.

[16] D. Brockhoff and E. Zitzler, "Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods," in *Proc. IEEE CEC*, 2007, pp. 2086–2093.

[17] A. López Jaimes, C. Coello Coello, and D. Chakraborty, "Objective reduction using a feature selection technique," in *Proc. GECCO*, 2008, pp. 673–680.

[18] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations," in *Proc. IEEE Int. Conf. SMC*, 2009, pp. 1758–1763.

[19] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature-PPSN VIII*. Berlin, Germany: Springer, 2004, pp. 832–842.

[20] J. Bader and E. Zitzler, "A hypervolume-based optimizer for high-dimensional objective spaces," in *New Developments in Multiple Objective and Goal Programming* (Lecture Notes in Economics and Mathematical Systems Series), D. Jones, M. Tamiz, and J. Ries (Eds.) Berlin, Heidelberg, Germany: Springer, 2010, vol. 638, pp. 35–54.

[21] E. Hughes, "Many-objective directed evolutionary line search," in *Proc. GECCO*, 2011, pp. 761–768.

[22] H. Sato, H. Aguirre, and K. Tanaka, "Controlling dominance area of solutions and its impact on the performance of MOEAs," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2007, pp. 5–20.

[23] H. Aguirre and K. Tanaka, "Space partitioning with adaptive ε-ranking and substitute distance assignments: A comparative study on many-objective MNK-landscapes," in *Proc. GECCO*, 2009, pp. 547–554.

[24] M. Köppen, R. Vicente-Garcia, and B. Nickolay, "Fuzzy-Pareto-dominance and its application in evolutionary multi-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2005, pp. 399–412.

[25] M. Köppen and C. Veenhuis, "Multi-objective particle swarm optimization by fuzzy-Pareto-dominance meta-heuristic," *Int. J. Hybrid Intell. Syst.*, vol. 3, no. 4, pp. 179–186, 2006.

[26] O. Schütze, "A new data structure for the nondominance problem in multi-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2003, pp. 509–518.

[27] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich, "Symbolic archive representation for a fast nondominance test," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2007, pp. 111–125.

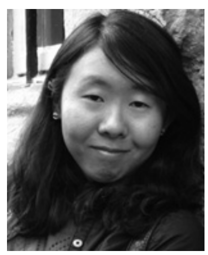
[28] H. Singh, A. Isaacs, and T. Ray, "A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems," *IEEE Trans. Evolut. Comput.*, vol. 15, no. 4, pp. 539–556, Aug. 2011.

[29] M. Koppen, K. Yoshida, and K. Ohnishi, "A GRATIS theorem for relational optimization," in *Proc. 11th Int. Conf. HIS*, 2011, pp. 674–679.

[30] K. Deb and S. Tiwari, "Omni-optimizer: A procedure for single and multi-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2005, pp. 47–61.

[31] K. McClymont and E. Keedwell, "Deductive sort and climbing sort: New methods for non-dominated sorting," *Evolut. Comput.*, vol. 20, no. 1, pp. 1–26, 2012.

[32] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. IEEE CEC*, 2002, pp. 825–830.



Handing Wang (S'10) received the B.Eng. degree in intelligence science and technology from Xidian Univerisity, Xi'an, China, in 2010, where she is currently pursuing the Ph.D. degree in circuit and systems.

She is currently a member of the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University. Her current research interests include nature-inspired computation and multiobjective optimization.

Xin Yao (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC, in 1990.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, and from 1990 to 1992, he was a Post-Doctoral Fellow with Australian National University, Canberra, Australia, and Commonwealth Scientific and Industrial Research Organization, Melbourne, Australia, and a Lecturer, Senior Lecturer, and Associate Professor with the University of New South Wales, Australian Defence Force Academy (ADFA), Canberra, from 1992 to 1999. Since April 1999, he has been a Professor (Chair) of computer science with the University of Birmingham, Birmingham, U.K., where he is currently the Director of the Centre of Excellence for Research in Computational Intelligence and Applications.

Dr. Yao is an IEEE CIS Distinguished Lecturer and a former EIC of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was a recipient of the 2001 IEEE Donald G. Fink Prize Paper Award, the 2010 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, the 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award, and several other Best Paper Awards.