Complex Intell. Syst. (2017) 3:247–263
DOI 10.1007/s40747-017-0057-5

ORIGINAL ARTICLE

Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization

Ye Tian[1] · Handing Wang[2] · Xingyi Zhang[1] · Yaochu Jin[2]

Received: 3 August 2017 / Accepted: 12 September 2017 / Published online: 23 September 2017

Abstract Since non-dominated sorting was first adopted in NSGA in 1995, most evolutionary algorithms have employed non-dominated sorting as one of the major criteria in their environmental selection for solving multi- and many-objective optimization problems. In this paper, we focus on analyzing the effectiveness and efficiency of non-dominated sorting in multi- and many-objective evolutionary algorithms. The effectiveness of non-dominated sorting is verified by considering two popular evolutionary algorithms, NSGA-II and KnEA, which were designed for solving multi- and many-objective optimization problems, respectively. The efficiency of non-dominated sorting is evaluated by comparing several state-of-the-art non-dominated sorting algorithms for multi- and many-objective optimization problems. These results provide important insights to adopt non-dominated sorting in developing novel multi- and many-objective evolutionary algorithms.

Keywords Non-dominated sorting · Evolutionary algorithm · Multi-objective optimization · Many-objective optimization

✉ Xingyi Zhang
xyzhanghust@gmail.com

Ye Tian
field910921@gmail.com

Handing Wang
handing.wang@surrey.ac.uk

Yaochu Jin
yaochu.jin@surrey.ac.uk

1 Institute of Bio-inspired Intelligence and Mining Knowledge, School of Computer Science and Technology, Anhui University, Hefei 230601, China

2 Department of Computer Science, University of Surrey, Guildford GU2 7XH, UK

Introduction

Multi-objective optimization problems (MOPs) refer to those consisting of multiple contradictory objectives to be optimized simultaneously, which widely exist in real-world applications [1–5]. MOPs with more than three objectives are also called many-objective optimization problems (MaOPs) [6]. Due to the fact that the objectives are in conflict with each other, there usually exists a set of trade-off solutions instead of one single optimal solution for an MOP. In the past 2 decades, evolutionary algorithms have demonstrated the superiority in solving MOPs, and a large number of multi-objective evolutionary algorithms (MOEAs) have been developed, such as NSGA-II [7], SPEA2 [8], PESA-II [9], IBEA [10], and MOEA/D [11].

For solving MOPs, one of the most important problems that should be addressed is how to distinguish the quality of solutions consisting of multiple objective values. To this end, Goldberg [12] suggested the use of the Pareto dominance [13] to sort a set of solutions for MOPs, and the sorting procedure is called non-dominated sorting. Briefly, the non-dominated sorting aims to divide a solution set into a number of disjoint subsets or ranks, by means of comparing their values of the same objective. After non-dominated sorting, solutions in the same rank are viewed equally important, and solutions in a smaller rank are better than those in a larger rank. Since NSGA [14] first adopted non-dominated sorting in 1995, it has become a widely adopted strategy in MOEAs. For non-dominated sorting, the effectiveness and efficiency are two important issues that have been concerned in MOEAs.

For MOPs, the effectiveness of non-dominated sorting has long been recognized and most of the existing MOEAs adopted this strategy, e.g., NSGA-II [7], PESA-II [9], GDE3 [15], SMPSO [16], EAG-MOEA/D [17], MOEA/IGD-NS [18], etc. In the existing MOEAs, non-dominated sorting is mainly performed in environmental selection, where solutions are divided into several ranks by non-dominated sorting and those in the same rank are distinguished by additional criteria, such as crowding distance in NSGA-II, GDE3, SMPSO, and EAG-MOEA/D, the region-based metric in PESA-II, and the enhanced IGD in MOEA/IGD-NS. As for MaOPs, non-dominated sorting has also been favored by researchers in developing MOEAs despite that its effectiveness considerably deteriorates on MaOPs due to the Pareto dominance resistance phenomenon [19]. Some representative MOEAs that directly adopted non-dominated sorting to handle MaOPs include GrEA [20], NSGA-III [21], KnEA [22], and LMEA [23]. There are also some recent works reported that non-dominated sorting is also an effective strategy to which did not use non-dominated sorting in their original versions, e.g., decomposition-based MOEAs with dominance, MOEA/DD [24], and BCE-MOEA/D [25].

The efficiency of non-dominated sorting is another issue in MOEAs, since it is often criticized due to the high computational cost. Taking the NSGA-II as an example, non-dominated sorting consumes more than 70% of the runtime in NSGA-II when a population size of 1000 and a maximum generation of 500 are adopted to solve a 2-objective DTLZ1. The computational cost will become much higher for larger population size or/and larger number of objectives. To address this issue, a lot of non-dominated sorting algorithms have been developed to improve the efficiency of non-dominated sorting, e.g., Jensen's sort [26], non-dominated ranking approach [27], deductive sort [28], and efficient non-dominated sort (ENS) [29]. There are also some non-dominated sorting methods specially tailored for solving MaOPs, such as corner sort [30], T-ENS [23], and A-ENS [31].

Despite that the effectiveness and efficiency of non-dominated sorting were widely concerned in the past years, a little work has been reported to analyze them in MOEAs, especially for solving MaOPs. In this paper, we empirically investigate the effectiveness and efficiency of non-dominated sorting in MOEAs for both MOPs and MaOPs. We verify the effectiveness of non-dominated sorting by considering two popular MOEAs developed for solving MOPs and MaOPs, NSGA-II and KnEA. Some rectifications to enhance the effectiveness of non-dominated sorting for many-objective optimization are also discussed. The efficiency of non-dominated sorting is evaluated by comparing eight existing non-dominated sorting algorithms in solving MOPs and MaOPs. These results will be helpful for researchers to develop new non-dominated sorting-based MOEAs which are both effective and efficient.

The rest of the paper is organized as follows. In "Basic concepts of multi-objective optimization and non-dominated sorting" section, some basic concepts of multi-objective optimization and non-dominated sorting are given. In "Effectiveness of non-dominated sorting for multi- and many-objective optimization" section, the effectiveness of non-dominated sorting is analyzed with some discussions on the existing rectifications for many-objective optimization. Afterwards, in "Efficiency of non-dominated sorting for multi- and many-objective optimization" section, the efficiency of non-dominated sorting is investigated by comparing several non-dominated sorting methods in dealing with MOPs and MaOPs. Finally, the conclusion is given in "Conclusion" section.

Basic concepts of multi-objective optimization and non-dominated sorting

A minimization MOP is defined as follows:

$$\begin{array}{l} \min \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ \text{s.t. } \mathbf{x} \in \Omega \end{array}, \tag{1}$$

where $\mathbf{x}$ is a solution in the decision space Ω, and $\mathbf{f}(\mathbf{x})$ contains M conflicting objectives to be minimized. A solution $\mathbf{x}$ is said to Pareto dominate solution $\mathbf{y}$, denoted as $\mathbf{x} \prec \mathbf{y}$, if and only if

$$\begin{cases} \forall i \in 1, \ldots, M: \ f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \\ \exists j \in 1, \ldots, M: \ f_j(\mathbf{x}) < f_j(\mathbf{y}) \end{cases}. \tag{2}$$

For an MOP, a Pareto optimal solution refers to a solution which is not dominated by any solution of the MOP, and all these solutions constitute the Pareto optimal solution set of the MOP. The projection of Pareto optimal solution set in objective space is called the Pareto front. Due to the conflicting nature of the objectives, there exists more than one Pareto optimal solution for an MOP. In particular, for an M-objective continuous MOP, its Pareto optimal solutions constitute an $M-1$ dimensional piecewise manifold [32].

Based on the Pareto dominance, solutions in a population P can be divided into L disjoint subsets or ranks $P = \{F_1, \ldots, F_L\}$, where L is the maximum number of subsets for population P. Non-dominated sorting is a procedure for finding these disjoint subsets, which usually consists of the following three steps:

- Step 1) Initialize the index i to 1.
- Step 2) Find all solutions which are not dominated by any solution in P and move them from P to F_i; $i = i + 1$.
- Step 3) If P is empty, stop; otherwise, go to Step 2.

Figure 1 depicts an example of non-dominated sorting, where the population is divided into three ranks. As shown in the figure, the following three conditions hold for non-dominated sorting:

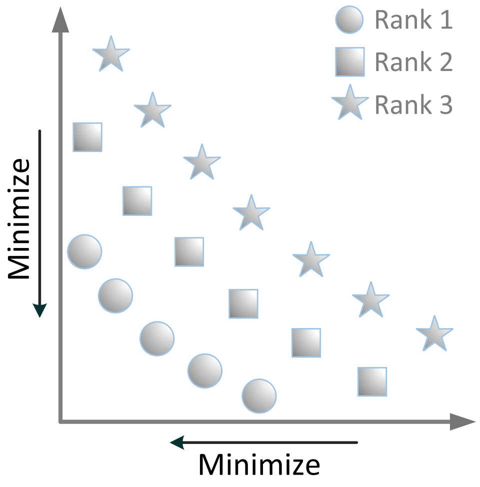


Fig. 1 Illustrative example of non-dominated sorting, where the population is divided into three ranks

$$\forall \mathbf{x}, \mathbf{y} \in F_j (1 \le j \le L), \quad \mathbf{x} \nprec \mathbf{y}, \tag{3}$$

$$\forall \mathbf{x} \in F_1, \forall \mathbf{y} \in P, \mathbf{y} \nprec \mathbf{x}, \tag{4}$$

$$\forall \mathbf{x} \in F_j, \exists \mathbf{y} \in F_{j-1} (2 \le j \le L), \mathbf{y} \prec \mathbf{x}. \tag{5}$$

To be specific, any solution in rank F_j, $1 \le j \le L$, cannot dominate another solution in this rank, i.e., all solutions in each rank are non-dominated mutually; the solutions in the first rank F_1 cannot be dominated by any solution in the population P; any solution in one rank is dominated by at least one solution in the former rank.

With non-dominated sorting, the quality of solutions in a population can be considerably distinguished, and this strategy has been widely adopted in MOEAs. In the next two sections, we will discuss the effectiveness and efficiency of non-dominated sorting in MOEAs, respectively.

Effectiveness of non-dominated sorting for multi- and many-objective optimization

Effectiveness for multi-objective optimization

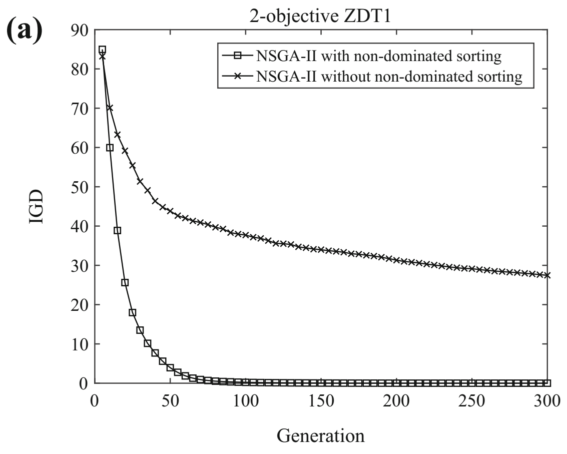
To verify the effectiveness of non-dominated sorting for multi-objective optimization, we empirically investigate the role of non-dominated sorting in NSGA-II, which is for solving MOPs [7]. The NSGA-II consists of two main components, fast non-dominated sort and crowding distance, for distinguishing the quality of solutions in the environmental selection. At each generation of NSGA-II, non-dominated sorting is first employed to select solutions with lower ranks from the population combining parent population with offspring population, and crowding distance is used as the secondary metric to distinguish solutions in the same rank by favoring solutions with a large crowding distance.

Figure 2 presents the convergence profiles of inverted generational distance (IGD) values obtained by NSGA-II with non-dominated sorting and the variant without non-dominated sorting on 2-objective ZDT1 and ZDT4, and 3-objective DTLZ1 and DTLZ2, averaged over 30 independent runs, where the population size is set to 100 and the remaining parameters are set to the same values recommended in [7]. IGD is a popular metric for measuring the quality of a solution set in terms of both convergence and distribution. ZDT1, ZDT4, DTLZ1, and DTLZ2 are widely used benchmark MOPs with multi- and uni-modal properties, respectively [33,34]. From the figure, the following two observations can be obtained.

First, non-dominated sorting plays a crucial role in guiding the population of NSGA-II to approximate the Pareto fronts of MOPs. On both uni- and multi-model MOPs, the population of NSGA-II is far from converging to the Pareto fronts in a maximum of 300 generations without using non-dominated sorting. Second, the crowding distance is a little helpful for the convergence of populations of NSGA-II on 2-objective MOPs, but the population cannot converge to the Pareto fronts of MOPs without non-dominated sorting. On 3-objective MOPs, it seems that the crowding distance cannot promote populations of NSGA-II towards the Pareto fronts. As the number of generations increases, the IGD values obtained by the NSGA-II without non-dominated sorting increase on 3-objective DTLZ1 and DTLZ2. The second observation can be confirmed by Fig. 3, where the minimal distance of solutions in population to Pareto fronts is presented at different iterations for the variant of NSGA-II without non-dominated sorting on 2-objective ZDT1 and ZDT4 and 3-objective DTLZ1 and DTLZ2.

To further illustrate the role of non-dominated sorting in MOEAs, Fig. 4 presents the number of solutions in different fronts determined by non-dominated sorting for next population of NSGA-II on 2-objective ZDT1 and ZDT4 and 3-objective DTLZ1 and DTLZ2, where the number of candidate solutions is 200 and the size of next population is 100. From the figure, it can be seen clearly that non-dominated sorting is a very effective strategy to distinguish the quality of solutions in population evolution of NSGA-II for solving MOPs. For the combined population consisting of parent and offspring populations, non-dominated sorting can determine a large number of candidate solutions unsuitable for surviving for next population when NSGA-II is used to solve MOPs. It can also be found that the number of solutions which are considered unsuitable for surviving considerably decreases on 3-objective MOPs, which implies the decrement of ability of non-dominated sorting in distinguishing the quality of solutions as the number of objectives increases.

Based on the above empirical results, we can conclude that non-dominated sorting is a promising strategy enabling the population to converge to the Pareto fronts for MOEAs.



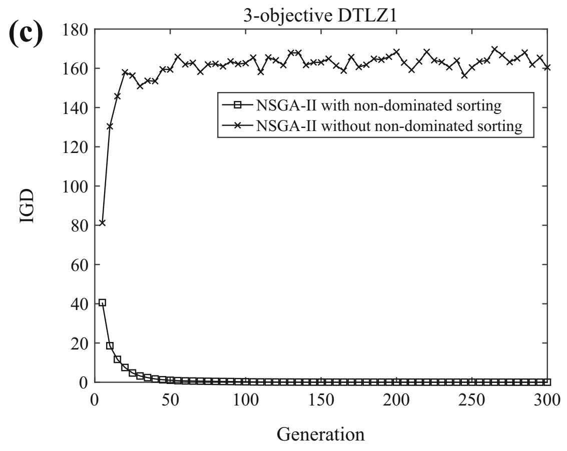


Fig. 2 Convergence profiles of IGD values obtained by NSGA-II with non-dominated sorting and the variant without non-dominated sorting on 2-objective ZDT1, 2-objective ZDT4, 3-objective DTLZ1, and 3-objective DTLZ2, averaged over 30 independent runs

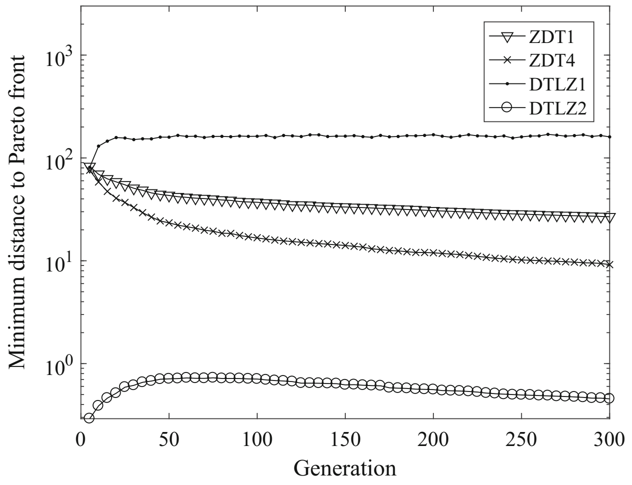


Fig. 3 Minimal distance of solutions in population to Pareto fronts during the population evolution of the variant of NSGA-II without non-dominated sorting on 2-objective ZDT1 and ZDT4 and 3-objective DTLZ1 and DTLZ2

Effectiveness for many-objective optimization

In this subsection, we evaluate the effectiveness of non-dominated sorting in MOEAs for solving MaOPs. To this end, we consider the knee point driven non-dominated sorting-based MOEA, KnEA, recently tailored to handle MaOPs. The KnEA consists of two main components in the environmental selection: (1) non-dominated sorting and (2) knee point selection. At each generation of KnEA, non-dominated sorting is first performed on the combined population consisting of parent and offspring populations, and then, knee points in the first non-dominated front are selected, in case the number of non-dominated solutions in the first front is larger than the population size.

Figure 5 presents the convergence profiles of IGD values obtained by KnEA with non-dominated sorting and the variant without non-dominated sorting on 5- and 10-objective DTLZ1 and DTLZ2, where the population size is set to 100 and the expected rate of knee points is set to 0.1 for DTLZ1 and 0.5 for DTLZ2. From the figure, the following three observations can be obtained. First, non-dominated sorting

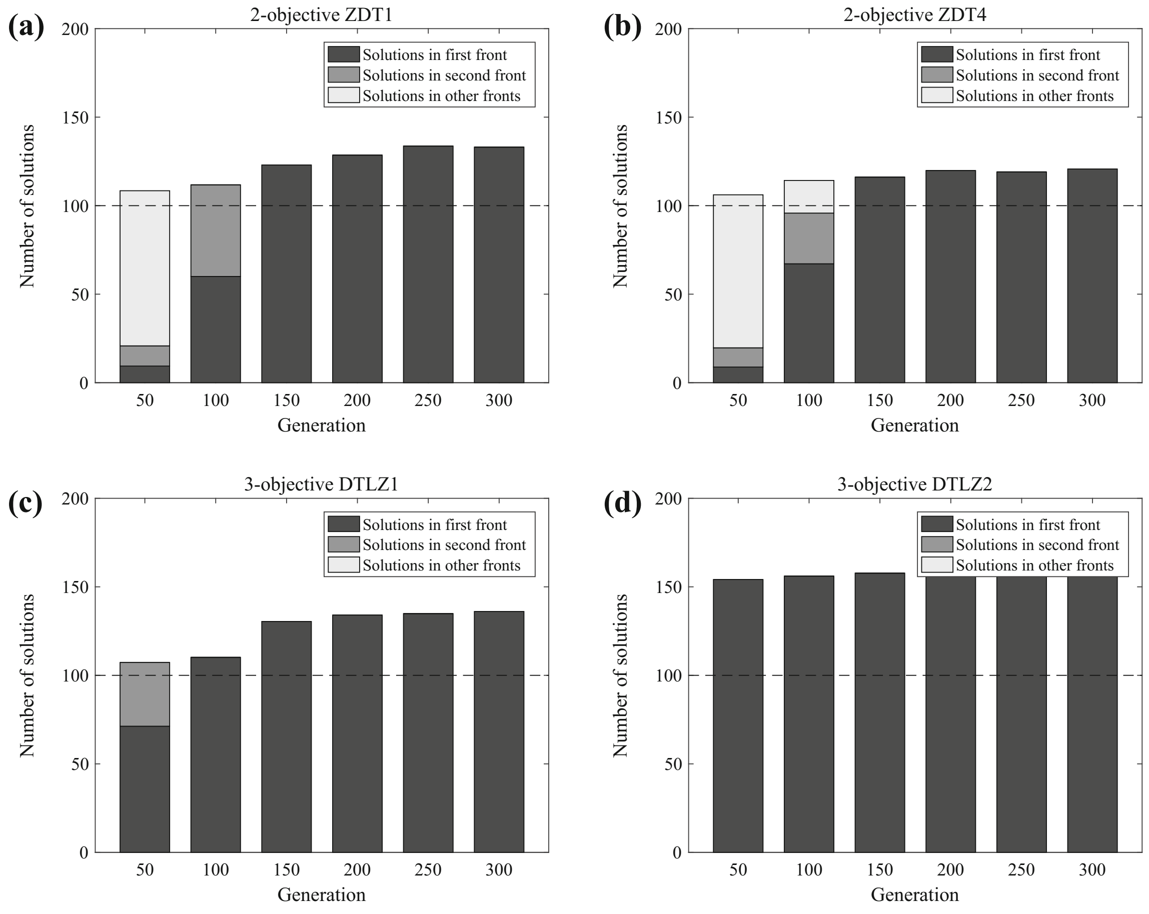


Fig. 4 Number of solutions in different fronts determined by non-dominated sorting for next population of NSGA-II on 2-objective ZDT1 and ZDT4 and 3-objective DTLZ1 and DTLZ2, where the population size is set to 100

is very helpful for KnEA to achieve a set of non-dominated solutions with better quality for solving MaOPs. The KnEA with non-dominated sorting can always obtain better IGD values than the variant of KnEA without non-dominated sorting under different iterations, on all tested MaOPs, especially for 5-objective DTLZ2. To illustrate the reason for the enhanced quality of solution set, Fig. 6 gives the generational distance (GD) values obtained by KnEA with non-dominated sorting and the variant without non-dominated sorting at different iterations on DTLZ1 and DTLZ2 with 5 and 10 objectives. The GD is a widely used metric for measuring the convergence of a solution set. As shown in the figure, non-dominated sorting can clearly help population of KnEA to better converge to the Pareto fronts on MaOPs, especially for DTLZ1 which has a large number of local Pareto fronts.

Second, the role of non-dominated sorting in promoting population of KnEA to converge to the Pareto fronts degenerates on MaOPs in comparison to that on MOPs. The main reason is attributed to a phenomenon called dominance resistance [19], since the number of solutions will considerably increase as the number of objectives increases. Taking two random solutions in M-dimensional objective space as an example, the probability that one solution dominates the other one is $(\frac{1}{2})^{M-1}$, as shown in Fig. 7. It is clear that the probability decreases rapidly with the increasing number of objectives, and it becomes almost impossible that solutions with more than 12 objectives in a random population can be dominated [35]. It is necessary to note that non-dominated sorting can still determine a few candidate solutions in the combined population unsuitable for surviving for next population in solving MaOPs, which is helpful for promoting population of KnEA to converge to the Pareto fronts.

Third, knee point selection plays a key role in enabling the population of KnEA to converge to the Pareto fronts for MaOPs. On all tested MaOPs, KnEA only using knee point selection can achieve a competitive performance in terms of IGD and GD, especially for DTLZ2 with 5 and 10 objectives. This result shows that KnEA is still effective on most MaOPs when the knee points are selected from the whole population instead of each non-dominated front of the population, despite that the performance has a little deterioration. It seems that non-dominated sorting is more helpful in KnEA

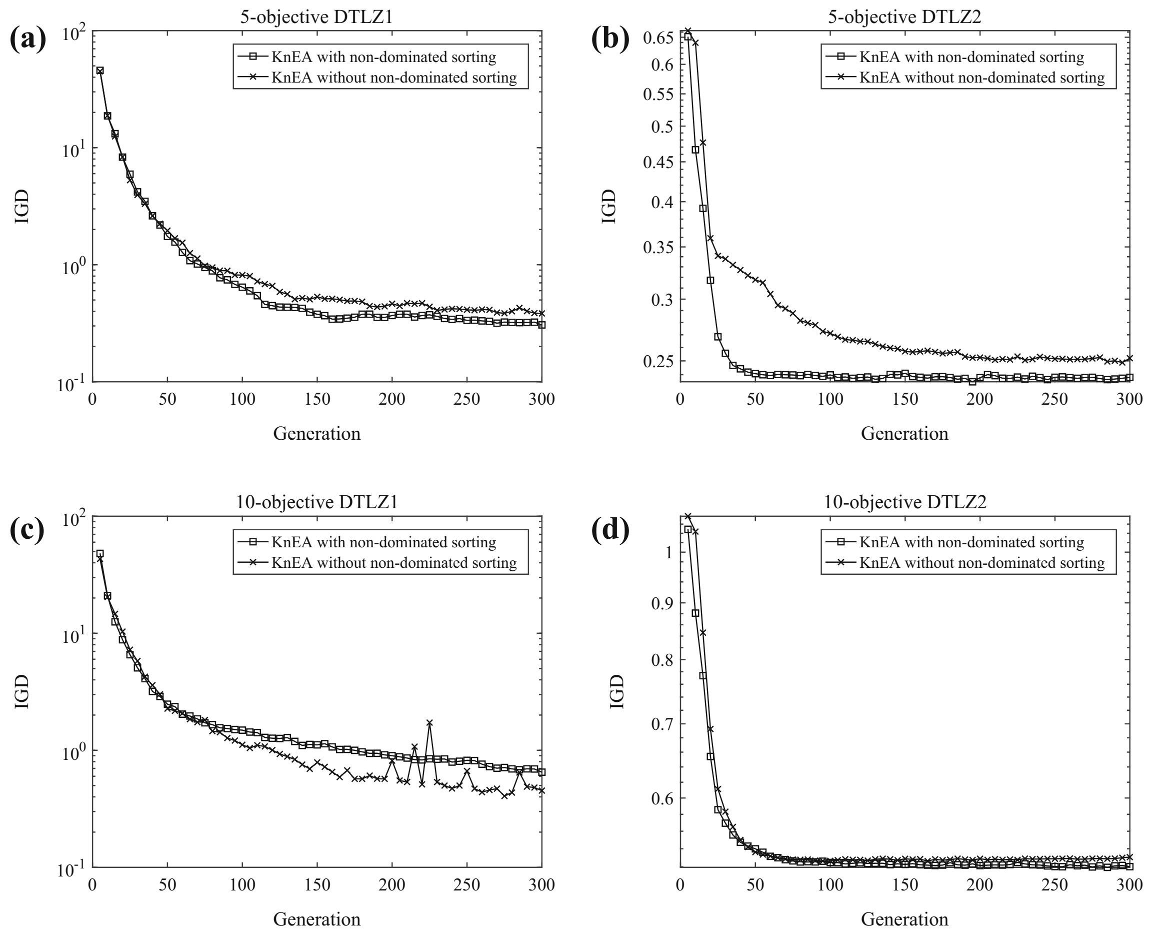


Fig. 5 Convergence profiles of IGD values obtained by KnEA with non-dominated sorting and the variant without non-dominated sorting on DTLZ1 and DTLZ2 with 5 and 10 objectives, averaged over 30 independent runs

on DTLZ1 with a large number of local Pareto fronts, which can considerably enhance the convergence of population of KnEA in terms of GD.

Therefore, we can summarize that non-dominated sorting is also important for developing a promising MOEA to solve MaOPs, especially for some complex MaOPs, such as those with multiple local Pareto fronts.

Rectifications of Pareto dominance for many-objective optimization

As shown in the above subsection, the effectiveness of non-dominated sorting degenerates in MOEAs for MaOPs due to the decrement of selection pressure of the Pareto dominance. To address this issue, a large number of enhanced versions of the Pareto dominance have been proposed based on different ideas.

The first idea for enhancing the effectiveness of the Pareto dominance is to divide the objective space into a number of grids and solutions in the same grid are considered as identical ones. One of representative dominance belonging to this category is ϵ-dominance relation [36]. ϵ-Dominance first divides each objective into d equal parts, and thus, an M-dimensional objective space will contain d^M hypercubes. Then, the grid coordinates of solutions are used to determine their dominance relations instead of the objective values. The ϵ-dominance relation is a relaxation of the Pareto dominance and other similar work includes $pa\epsilon$-dominance [37], cone ϵ-dominance [38], etc.

The second idea for enhancing the effectiveness of the Pareto dominance is based on the expansion of the dominance area. For two random solutions with M objectives, the probability that one solution dominates the other one is $(\frac{1}{2})^{M-1}$, and hence, it is rare for one solution to dominate the other one in high-dimensional space. The controlling dominance area of solutions (CDAS) method [39] expands the dominance area of each solution by a specified angle on each objective, and thus, the probability of dominance and the selection pressure increase. An adaptive version of CDAS was suggested in [40], where the expanding angle was adaptively estimated according to the extreme solutions. It is worth noting that the CDAS expands the dominance area by

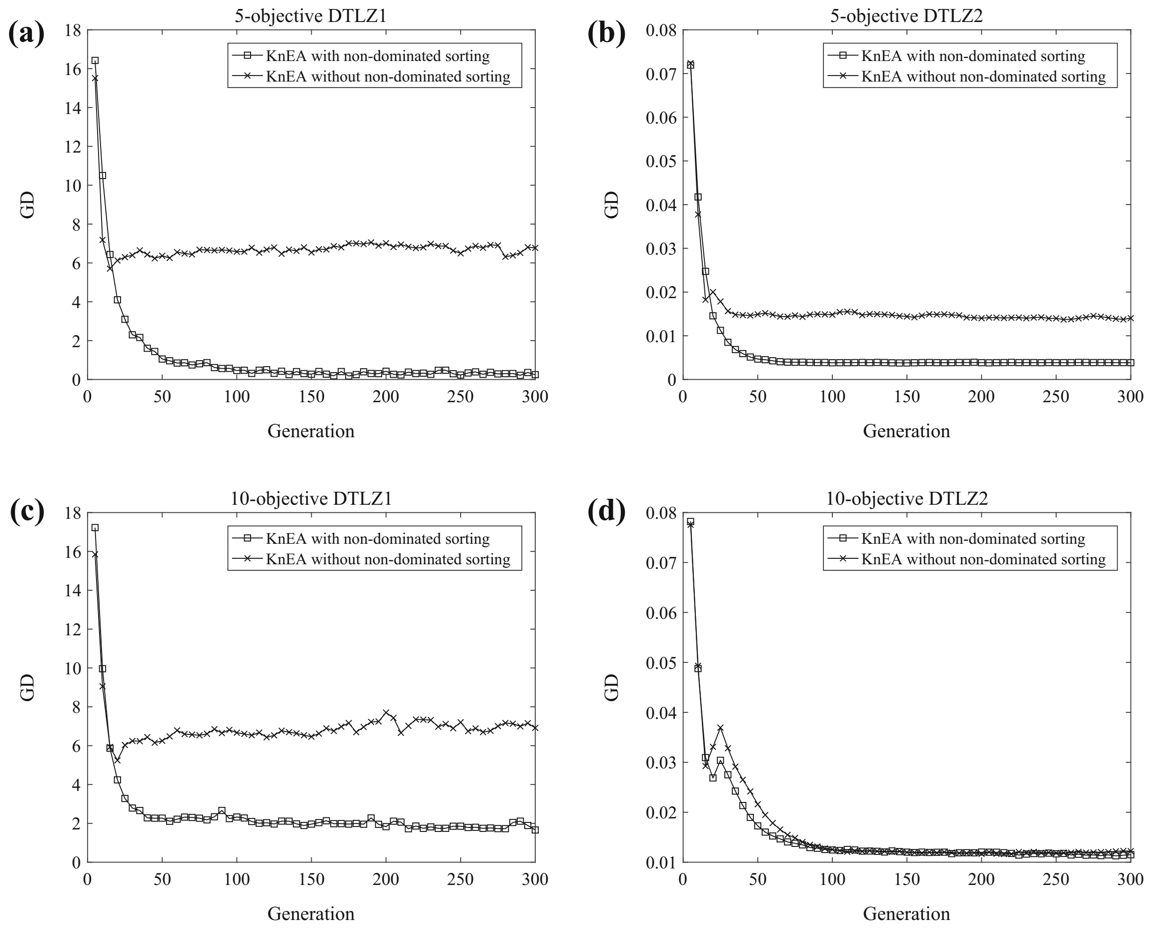


Fig. 6 Convergence profiles of GD values obtained by KnEA with non-dominated sorting and the variant without non-dominated sorting on DTLZ1 and DTLZ2 with 5 and 10 objectives, averaged over 30 independent runs

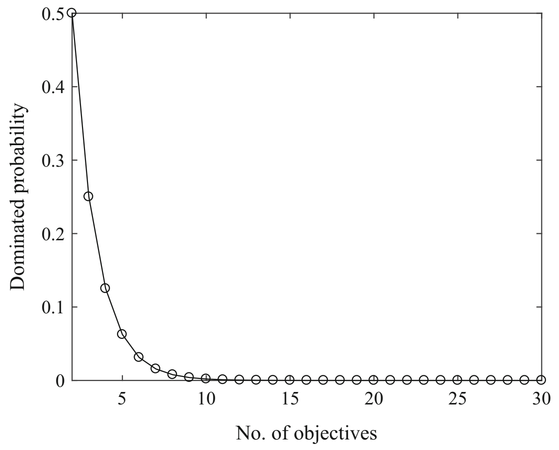


Fig. 7 Probability of one random solution dominating another one for different numbers of objectives

modifying the objective values of solutions; there also exist some dominance relations which expand the dominance area by modifying the definition of dominance, e.g., α-dominance [41] and generalized Pareto optimality (GPO) [42].

The third idea is to adopt the concept of fuzzy logic to develop novel dominance relations, such as (1-k)-dominance [43], L-dominance [44], and fuzzy dominance [45,46]. In the Pareto dominance, one solution dominates another one only if all the objective values of the former are smaller than or equal to those of the latter, whereas in fuzzy logic-based dominance relations, one solution may dominate another one if the majority of the objectives of the former are smaller than those of the latter. In this case, a solution can dominate those which have much worse values than it on most objectives and slightly better values than it on a few objectives, and thus, the quality of solutions can be distinguished.

The fourth idea enhances the effectiveness of the Pareto dominance by means of a set of uniformly distributed reference vectors as suggested in decomposition-based MOEAs [47,48]. θ-Dominance [48] is a dominance relation belonging to this category, where each solution is associated with its nearest reference vector, and a solution is said to dominate another one if and only if the two solutions are associated with the same reference vector and the former has better convergence and diversity than the latter. θ-dominance relation aims to make each solution converge to the same direction of one reference vector, which can enable the population to hold a good convergence and diversity.

It is necessary to stress that there are also some other interesting ideas which enhance the effectiveness of non-dominated sorting by combining it with additional convergence- or diversity-related metrics, instead of directly modifying the definition of the Pareto dominance. Some representatives belonging to this category include hypervolume [49], knee point [22], enhanced IGD [18], and shift-based density [50].

Efficiency of non-dominated sorting for multi- and many-objective optimization

Main non-dominated sorting methods and their time complexity

In the past 2 decades, a large number of interesting algorithms have been developed to address the high computational cost of non-dominated sorting. In what follows, we only recall several non-dominated sorting algorithms which are widely used in literature. The interested readers can refer to [51] for a more detailed list of non-dominated sorting algorithms.

In 2003, Jensen [26] suggested a non-dominated sorting method, called Jensen's sort, based on the divide-and-conquer strategy. Jensen's sort is an improved version of the non-dominated sorting method developed by Kung et al. [52], where the time complexity has been reduced to $O(N \ln^{M-1} N)$ from $O(N^2 \ln^{M-1} N)$ (M and N are hereafter denoted as the number of objectives and the population size, respectively), considerably outperforming the time complexity $O(MN^2)$ of fast non-dominated sort (FNS) developed in NSGA-II. Compared to the FNS, Jensen's sort also has a significant improvement in space complexity. The space complexity of Jensen's sort is $O(N)$, whereas FNS holds a space complexity of $O(N^2)$. Hence, Jensen's sort is computationally more efficient than FNS on MOPs with a small number of objectives, especially for those with two or three objectives. In spite of the high computational efficiency, Jensen's sort suffers from the restriction that two compared solutions should not have the same value in any of their objectives.

To address the weakness of Jensen's sort, Fang et al. [53], in 2008, proposed a new divide-and-conquer-based non-dominated sorting algorithm, where a new data structure called dominance tree was adopted to reduce the number of redundant comparisons in FNS. This algorithm is called the divide-and-conquer based sort. Empirical results demonstrated that the divide-and-conquer-based sort holds a time complexity close to $O(MN \ln N)$ for two-objective MOPs and approaches asymptotically to the upper bound $O(MN^2)$ as the number of objectives increases. Another improved version of Jensen's sort, termed generalized Jensen's sort, was also developed by Fortin et al. in [54]. As reported in [54], the generalized Jensen's sort can well address the weakness of Jensen's sort without increasing the time complexity and space complexity.

In 2012, McClymont and Keedwell [28] developed a novel non-dominated sorting method, called deductive sort, by exploiting the properties of Pareto optimality, dominance, and non-dominance, as well as the possible inherent inferences that can be made based on the nature of these relationships. Although deductive sort holds a time complexity of $O(MN^2)$ in the worst case which is the same as that of FNS, empirical evidence indicated that it can save a large number of comparisons between solutions.

The corner sort suggested by Wang and Yao [30] in 2013 is also a computationally very efficient non-dominated sorting algorithm. In corner sort, a non-dominated solution is first selected from the corner solutions, and then, the solutions dominated by it will be ignored to save comparisons between solutions. As reported in [30], corner sort is more suited for solving MaOPs than FNS and deductive sort, and the more objectives an MaOP has, the more objective comparisons it can save. Corner sort holds a time complexity of $O(MN^2)$ in the worst case, but in some best cases, its time complexity can be reduced to $O(MN\sqrt{N})$.

In 2015, Martin et al. [55] proposed a non-dominated sorting method, called M-front, whose main idea was to utilize the existing knowledge of population in dominance relationship at last generation to perform non-dominated sorting for the current population. In M-front, the geometric and algebraic properties of the Pareto dominance were used to speed up the insertions and removals of solutions in non-dominated fronts. The M-front has a best case time complexity of $O(MN)$ or $O(MN \ln N)$ if the k-d tree is used, and a worst case time complexity of $O(MN^2)$.

Zhang et al. [29] also developed an efficient non-dominated sorting method in 2015, called ENS, which has been shown to well suit for solving MOPs with a small number of objectives, especially for MOPs with two or three objectives. The high efficiency of ENS is attributed to the fact that an operation of pre-sort was suggested for population according to one of the objectives, since in the sorted population, a solution can never be dominated by solutions ranked behind it, thereby saving a large number of comparisons between solutions. It is worth noting that the superiority of ENS will decrease as the number of objectives increases, despite that it still outperforms several existing non-dominated sorting methods, such as FNS and deductive sort as indicated in [29]. To solve this problem, a tree-based non-dominated sorting method, termed T-ENS, was suggested by Zhang et al. in [23] based on the ENS framework. Empirical evaluations demonstrated that T-ENS is computationally very efficient for MaOPs and

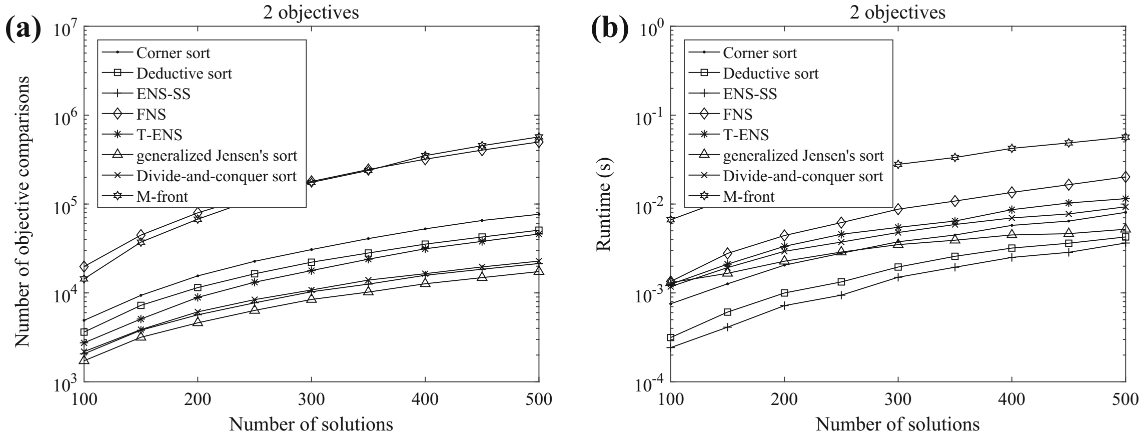


Fig. 8 Number of objective comparisons and runtime(s) of the eight compared non-dominated sorting methods on random populations with different sizes for two-objective optimization

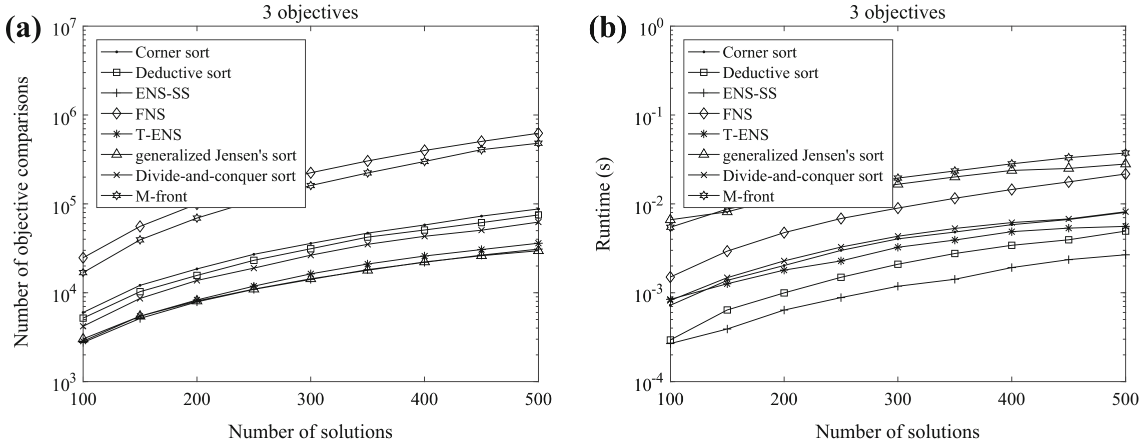


Fig. 9 Number of objective comparisons and runtime(s) of the eight compared non-dominated sorting methods on random populations with different sizes for three-objective optimization

its computational cost almost keeps the same as the number of objectives increases. T-ENS holds a worst case time complexity of $O(MN^2)$ (the same with that of ENS) and a best case time complexity of $O(MN\ln N/\ln M)$, which is better than $O(MN\ln N)$ of ENS-BS and $O(MN\sqrt{N})$ of ENS-SS.

Recently, an interesting non-dominated sorting approach for many-objective optimization, called A-ENS, was reported in [31], where the idea of approximate non-dominated sorting was developed. The main difference between A-ENS and existing non-dominated sorting methods lies in the fact that A-ENS can only obtain an approximate non-dominated sorting result, whereas the other methods all aim to obtain an accurate result. Empirical validation by embedding it into three popular MOEAs showed that A-ENS is not only computationally very efficient for MaOPs, but also can improve search performance on most test problems. The time complexity of A-ENS is $O(N^2)$ in the worst case and $O(N\sqrt{N})$ in the best cases, which is independent of the number of objectives, since, in A-ENS, the dominance relationship between any two solutions is determined by a maximum of three-objective comparisons.

Efficient non-dominated sorting for multi-objective optimization

In the following, we verify the computational efficiency of eight widely used non-dominated sorting methods when they are adopted for handling MOPs. The eight methods under consideration include fast non-dominated sort (FNS) [7], generalized Jensen's sort [54], divide-and-conquer-based sort [53], deductive sort [28], corner sort [30], M-front [55], ENS-SS [29], and T-ENS [23].

In the experiments, the computational efficiency of these non-dominated sorting methods is considered on random populations and MOEAs. The first scenario is used to mimic the situation in the early search stages of MOEAs, and the second scenario is adopted to test the computational efficiency of these non-dominated sorting methods when they

Table 1 Number of objective comparisons of the eight non-dominated sorting methods when they are embedded into NSGA-II for solving two-objective and three-objective DTLZ2

Methods	$N = 100$		$N = 500$	
	$M = 2$	$M = 3$	$M = 2$	$M = 3$
FNS	1.6E+7	1.9E+7	4.0E+8	4.7E+8
Generalized Jensen's sort	5.0E+5	**1.1E+6**	3.8E+6	**7.9E+6**
Divide-and-conquer sort	5.6E+6	8.0E+6	1.2E+8	2.0E+8
Deductive sort	5.2E+6	7.7E+6	1.2E+8	1.9E+8
Corner sort	4.6E+6	6.6E+6	10.0E+7	1.5E+8
M-front	3.8E+6	5.1E+6	8.8E+7	1.1E+8
T-ENS	3.1E+6	1.5E+6	6.9E+7	2.0E+7
ENS-SS	**3.9E+5**	4.2E+6	**2.6E+6**	9.8E+7

The best result on each test instance is in bold

are embedded into MOEAs for solving MOPs. In the second scenario, all components of an embedded MOEA are identical with the only exception of the non-dominated sorting methods adopted in it.

Figures 8 and 9 present the number of objective comparisons and runtime(s) of the eight non-dominated sorting methods in the first scenario for two-objective and three-objective optimization, averaged over 30 random populations with the same size. From the figures, the following two results can be observed. First, in terms of number of objective comparisons, the generalized Jensen's sort performs the best on random populations for both two-objective and three-objective optimization. ENS-SS and divide-and-conquer-based sort need slightly more objective comparisons than generalized Jensen's sort. The rest five non-dominated sorting methods underperform the generalized Jensen's sort, ENS-SS, and divide-and-conquer-based sort.

Second, in terms of runtime, ENS-SS always achieves the best efficiency on random populations for two-objective and three-objective optimization. The superiority of ENS-SS over the generalized Jensen's sort in runtime may be partly attributed to the fact that ENS-SS holds a space complexity of $O(1)$, whereas the space complexity of generalized Jensen's sort is $O(N)$. It is worth noting that deductive sort also achieves a competitive efficiency on random populations in terms of runtime despite that it uses more objective comparisons.

Tables 1 and 2 list the experimental results of the eight non-dominated sorting methods when they are embedded into NSGA-II with a population size of 100 and 500 for solving two-objective and three-objective DTLZ2, averaged over 30 runs. All parameters of NSGA-II are set as recommended in [7]. From the tables, it can be found that the superiority of ENS-SS over the other seven non-dominated sorting methods has been enhanced when they are embedded into MOEAs to solve MOPs, in terms of both number of objective comparisons and runtime.

For the number of objective comparisons, the generalized Jensen's sort still performs the best for three-objective

Table 2 Ratio of runtime of the eight non-dominated sorting methods to that of ENS-SS when they are embedded into NSGA-II for solving two-objective and three-objective DTLZ2

Methods	$N = 100$		$N = 500$	
	$M = 2$	$M = 3$	$M = 2$	$M = 3$
FNS	18.98	4.18	93.84	4.59
Generalized Jensen's sort	16.98	21.13	16.73	4.82
Divide-and-conquer sort	14.02	2.91	41.41	2.39
Deductive sort	6.94	1.82	30.47	1.98
Corner sort	15.60	3.65	46.74	2.78
M-front	43.18	9.39	66.51	3.41
T-ENS	10.25	1.26	33.95	**0.51**
ENS-SS	**1.00**	**1.00**	**1.00**	1.00

The best result on each test instance is in bold

DTLZ2, but ENS-SS can achieve the best efficiency in solving two-objective DTLZ2. As for the runtime, ENS-SS takes much less cost than the other seven non-dominated sorting methods for solving two-objective and three-objective DTLZ2, in case a population size of 100 is used. When the population size increases to 500, ENS-SS takes the least runtime for solving two-objective DTLZ2, whereas T-ENS will achieve the best for three-objective DTLZ2.

From the tables, it can also be seen that the computational efficiency of ENS-SS will be enhanced when a large population size of NSGA-II is used to solve MOPs, especially for solving MOPs with two objectives. Take the well-known non-dominated sorting method FNS as a comparison, the runtime taken by FNS is about 19 times of that of ENS-SS if a population size of 100 is used to solve two-objective DTLZ2. This ratio will be incremented to 94 as the population size becomes 500.

On the basis of the above empirical comparisons, we can conclude that ENS-SS is computationally more efficient than the state-of-the-art non-dominated sorting methods for multi-objective optimization, which can significantly improve the computational efficiency of an MOEA when it is embedded into the algorithm to solve MOPs. In the case of solving

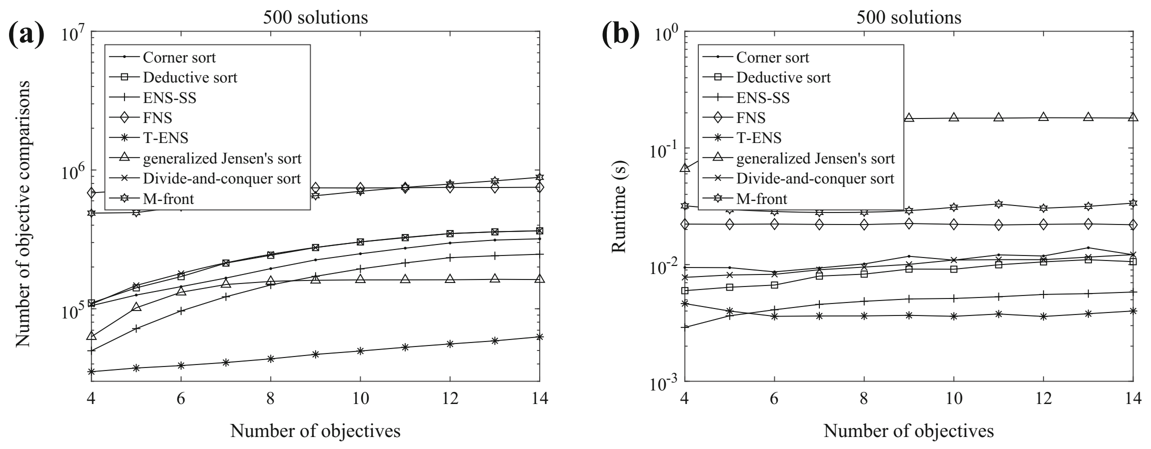


Fig. 10 Number of objective comparisons and runtime(s) of the eight compared non-dominated sorting methods on random populations with a size of 500 for many-objective optimization

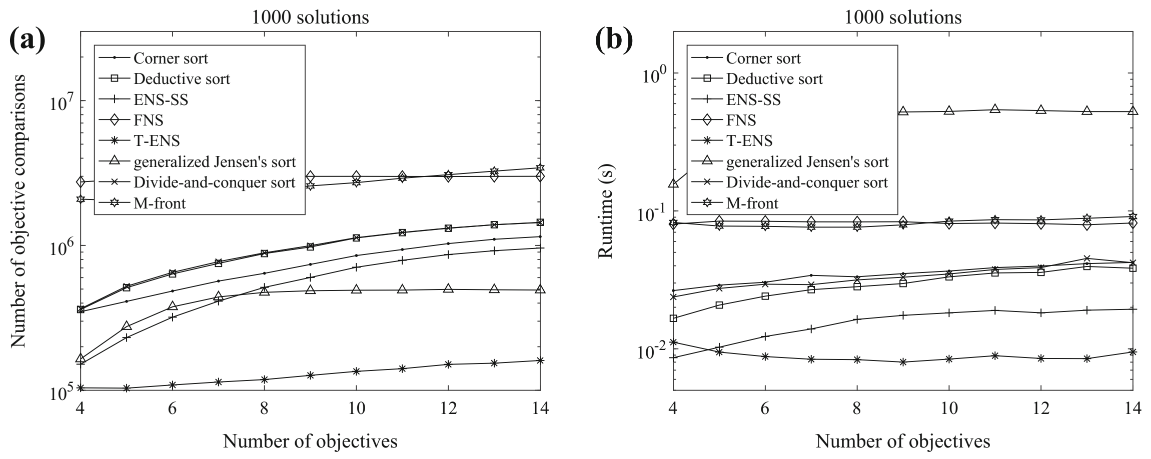


Fig. 11 Number of objective comparisons and runtime(s) of the eight compared non-dominated sorting methods on random populations with a size of 1000 for many-objective optimization

MOPs with three objectives, T-ENS will be strongly suggested to be adopted if a large population size is used in the MOEA.

Efficient non-dominated sorting for many-objective optimization

In this subsection, we test the computational efficiency of the above eight non-dominated sorting methods on MaOPs.

The experiments are conducted on three different scenarios which are often encountered in many-objective optimization. In the first scenario, we test the efficiency of the non-dominated sorting methods on random populations for many-objective optimization; in the second scenario, the computational costs of the non-dominated sorting methods are compared by embedding them into the MOEAs which are specially tailored to solve MaOPs; in the third scenario, we compare the efficiency of the non-dominated sorting methods when they are used to obtain a set of reference points uniformly distributed on the true PFs, which is required for calculating some performance indicators, such as GD [56] and IGD [57].

Figures 10 and 11 plot the experimental results of the eight non-dominated sorting methods in the first scenario averaged over 30 random populations for the same number of objectives, where two population sizes of 500 and 1000 are considered, respectively. From the figures, the following two conclusions can be obtained. First, the efficiency of T-ENS is superior over the other seven non-dominated sorting methods in terms of both number of objective comparisons and runtime on random populations for many-objective optimization. The superiority of T-ENS will be enhanced as the population size increases. Second, ENS-SS performs better than T-ENS on random populations with four objectives in terms of runtime, despite that it consumes more objective comparisons than T-ENS. The main reason is attributed to the fact that ENS-SS holds a space complexity of $O(1)$, whereas the space complexity of T-ENS is $O(N)$.

Table 3 Number of objective comparisons of the eight non-dominated sorting methods when they are embedded into KnEA for solving 5-objective and 10-objective DTLZ2

Methods	$N = 100$		$N = 500$	
	$M = 5$	$M = 10$	$M = 5$	$M = 10$
FNS	2.1E+7	2.8E+7	5.3E+8	6.6E+8
Generalized Jensen's sort	4.6E+6	6.7E+6	4.7E+7	8.8E+7
Divide-and-conquer sort	1.0E+7	1.4E+7	2.5E+8	3.1E+8
Deductive sort	1.0E+7	1.4E+7	2.5E+8	3.0E+8
Corner sort	9.2E+6	1.4E+7	1.9E+8	2.5E+8
M-front	1.2E+7	2.5E+7	2.3E+7	5.5E+8
ENS-SS	6.5E+6	1.0E+7	1.5E+8	2.1E+8
T-ENS	**1.7E+6**	**2.8E+6**	**1.8E+7**	**2.4E+7**

The best result on each test instance is in bold

Table 4 Ratio of runtime of the eight non-dominated sorting methods to that of T-ENS when they are embedded into KnEA for solving 5-objective and 10-objective DTLZ2

Methods	$N = 100$		$N = 500$	
	$M = 5$	$M = 10$	$M = 5$	$M = 10$
FNS	3.97	4.17	11.15	10.57
Generalized Jensen's sort	58.97	71.43	57.60	92.60
Divide-and-conquer sort	2.67	2.67	6.09	5.65
Deductive sort	1.91	2.06	5.27	4.90
Corner sort	3.66	3.51	7.27	6.08
M-front	10.53	13.20	11.10	14.13
ENS-SS	1.13	1.36	3.14	3.32
T-ENS	**1.00**	**1.00**	**1.00**	**1.00**

The best result on each test instance is in bold

Table 5 Number of objective comparisons of the eight non-dominated sorting methods for obtaining a set of reference points uniformly distributed on the true PF of DTLZ7 under different numbers of sampled points

Methods	1000 Points		10,000 Points	
	$M = 5$	$M = 10$	$M = 5$	$M = 10$
FNS	2.9E+6	3.0E+6	2.9E+8	3.0E+8
Generalized Jensen's sort	2.3E+5	**3.5E+5**	**5.4E+6**	**1.3E+7**
Divide-and-conquer sort	8.9E+5	1.5E+6	5.0E+7	1.4E+8
Deductive sort	7.5E+5	1.5E+6	3.3E+7	1.4E+8
Corner sort	5.9E+5	1.2E+6	2.7E+7	1.0E+8
M-front	9.0E+5	2.6E+6	5.4E+7	2.1E+8
ENS-SS	4.9E+5	1.0E+6	2.7E+7	9.4E+7
T-ENS	**1.7E+5**	3.7E+5	9.1E+6	2.5E+7

The best result on each test instance is in bold

Table 6 Ratio of runtime of the eight non-dominated sorting methods to that of T-ENS when they are used to obtain a set of reference points uniformly distributed on the true PF of DTLZ7 under different numbers of sampled points

Methods	1000 Points		10,000 Points	
	$M = 5$	$M = 10$	$M = 5$	$M = 10$
FNS	7.98	4.61	12.14	4.69
Generalized Jensen's sort	36.69	32.19	11.02	10.99
Divide-and-conquer sort	3.32	2.59	3.31	2.45
Deductive sort	2.40	2.35	1.88	2.28
Corner sort	2.99	2.94	2.59	3.01
M-front	5.95	6.77	4.04	4.82
ENS-SS	2.00	1.46	2.37	1.39
T-ENS	**1.00**	**1.00**	**1.00**	**1.00**

The best result on each test instance is in bold

Tables 3 and 4 list the experimental results of the eight non-dominated sorting methods when they are embedded into KnEA with a population size of 100 and 500 to solve 5-objective and 10-objective DTLZ2, respectively. The parameter settings of KnEA are the same as those recommended in [22] and the reported results are averaged over 30 independent runs. From the tables, we can find that T-ENS performs much more efficient than the other seven non-dominated sorting methods, in case that they are embedded into MOEAs to solve MaOPs. ENS-SS achieves the second best efficiency in terms of runtime when they are embedded into KnEA to solve MaOPs. It can also be seen that the superiority of T-ENS over the compared methods will be enhanced as the population size and the number of objectives increase in KnEA. These empirical results show that T-ENS is more suited to deal with large-scale MaOPs, since a large population is often needed for solving large-scale optimization problems.

Tables 5 and 6 present the experimental results of the eight non-dominated sorting methods for obtaining a set of reference points uniformly distributed on the true PF of DTLZ7 under 1000 and 10,000 sampled points, respectively. As can be seen from the tables, the generalized Jensen's sort achieves the fewest objective comparisons in obtaining a set of reference points uniformly distributed on the true PF of DTLZ7, except in the case of 1000 sampled points for 5-objective DTLZ7, where T-ENS performs the fewest objective comparisons. In terms of runtime, T-ENS takes the least among all eight considered non-dominated sorting methods, despite that it consumes more objective comparisons than the generalized Jensen's sort. It can also be found that ENS-SS is the second less time-consuming non-dominated sorting method except in the case of 10,000 sampled points for 10-objective DTLZ7, where deductive sort performs the second best.

From the empirical results shown in the above three scenarios, we can conclude that T-ENS is more suited for many-objective optimization, whose computational efficiency is much more competitive than that of the existing non-dominated sorting methods, especially when a large population size is adopted.

Approximate non-dominated sorting for many-objective optimization

In the above two subsections, we have empirically verified the computational efficiency of eight state-of-the-art non-dominated sorting algorithms for multi-objective and many-objective optimization, respectively. In this subsection, we consider another interesting idea of performing non-dominated sorting for many-objective optimization, called approximate non-dominated sorting. A-ENS was the first algorithm developed recently based on approximate non-dominated sorting for many-objective optimization [31].

Table 7 Ratio of runtime of A-ENS to that of T-ENS when they are embedded into KnEA and Two_Arch2 for solving DTLZ1–DTLZ7 and WFG1–WFG9 with 5 and 10 objectives

Problems	KnEA		Two_Arch2	
	$M = 5$	$M = 10$	$M = 5$	$M = 10$
DTLZ1	0.59	0.53	0.64	0.53
DTLZ2	0.82	0.68	0.75	0.71
DTLZ3	0.60	0.46	0.63	0.49
DTLZ4	0.74	0.68	0.79	0.76
DTLZ5	0.55	0.47	0.39	0.44
DTLZ6	0.54	0.56	0.43	0.52
DTLZ7	0.59	0.5	0.50	0.44
WFG1	0.60	0.56	0.66	0.55
WFG2	0.59	0.54	0.65	0.63
WFG3	0.65	0.58	0.52	0.50
WFG4	0.75	0.72	0.74	0.76
WFG5	0.76	0.69	0.79	0.81
WFG6	0.73	0.69	0.74	0.76
WFG7	0.76	0.71	0.79	0.81
WFG8	0.65	0.62	0.68	0.67
WFG9	0.69	0.72	0.80	0.77

The main difference between A-ENS and existing non-dominated sorting algorithms lies in the fact that, instead of identifying the accurate non-dominated sorting result for a given population in existing non-dominated sorting algorithms, A-ENS determines an approximate sorting result by performing at most three-objective comparisons for each pair of solutions. This means that the existing non-dominated sorting algorithms always find the same sorting result for a given population, and these algorithms distinguish themselves only in the computational efficiency. A-ENS obtains a sorting result different from that of the other non-dominated sorting algorithms due to the errors caused by approximate sorting. In the following, we empirically verify the efficiency of A-ENS and the influence on performance of MOEAs by embedding it into two MOEAs, KnEA and Two_Arch2, developed recently for solving MaOPs. All reported experimental results are obtained by averaging over 30 independent runs.

Table 7 presents the computational efficiency of A-ENS in KnEA and Two_Arch2 on DTLZ1–DTLZ7 [58] and WFG1–WFG9 [59] with 5 and 10 objectives, in comparison with the accurate non-dominated sorting algorithm T-ENS. The parameter settings of KnEA and Two_Arch2 are the same as recommended in [22,60]. From the table, it can be seen clearly that A-ENS is more efficient than T-ENS in MOEAs for solving MaOPs. Compared to the accurate non-dominated sorting T-ENS, A-ENS consumes roughly 70% runtime of that of T-ENS in both KnEA and Two_Arch2 to solve DTLZ

Table 8 HV values obtained by KnEA with accurate non-dominated sorting method and A-ENS on DTLZ1–DTLZ7 and WFG1–WFG9 with 5 and 10 objectives

Problems	$M = 5$		$M = 10$	
	T-ENS	A-ENS	T-ENS	A-ENS
DTLZ1	4.5969E−1 ± 8.82E−2	**5.0734E−1 ± 1.25E−1**	**2.7163E−1 ± 1.41E−1**	7.9948E−2 ± 1.02E−1
DTLZ2	**6.1657E−1 ± 6.15E−3**	6.0946E−1 ± 8.69E−3	8.1704E−1 ± 6.24E−2	**8.4099E−1 ± 1.18E−2**
DTLZ3	5.6254E−2 ± 1.04E−1	**2.6701E−1 ± 7.27E−2**	3.9095E−3 ± 1.26E−2	**6.3055E−2 ± 7.43E−2**
DTLZ4	**6.1858E−1 ± 6.89E−3**	6.1811E−1 ± 6.61E−3	8.0266E−1 ± 2.05E−2	**8.3580E−1 ± 1.66E−2**
DTLZ5	**7.4163E−2 ± 1.80E−2**	7.3073E−2 ± 2.46E−2	**3.8298E−2 ± 1.25E−2**	3.5344E−2 ± 2.08E−2
DTLZ6	2.4864E−2 ± 1.87E−2	**3.4327E−2 ± 2.08E−2**	**1.6162E−2 ± 2.29E−2**	2.8049E−4 ± 8.72E−4
DTLZ7	1.3594E−1 ± 6.57E−3	**1.4844E−1 ± 5.74E−3**	2.2713E−2 ± 8.14E−3	**9.9292E−2 ± 7.58E−3**
WFG1	9.7626E−1 ± 2.32E−2	**9.8972E−1 ± 8.03E−3**	9.4577E−1 ± 8.51E−2	**9.9397E−1 ± 2.66E−3**
WFG2	**9.7379E−1 ± 4.23E−2**	9.2167E−1 ± 9.42E−2	9.8899E−1 ± 1.93E−3	**9.9603E−1 ± 2.87E−3**
WFG3	5.3516E−1 ± 1.40E−2	**5.6173E−1 ± 2.11E−2**	5.3188E−1 ± 1.73E−2	**5.5059E−1 ± 1.33E−2**
WFG4	**5.6356E−1 ± 9.52E−3**	5.6327E−1 ± 7.44E−3	7.7241E−1 ± 2.30E−2	**7.9026E−1 ± 1.09E−2**
WFG5	**5.4986E−1 ± 5.80E−3**	5.4724E−1 ± 5.75E−3	**7.6269E−1 ± 5.32E−3**	7.5885E−1 ± 4.79E−3
WFG6	**5.3199E−1 ± 2.63E−2**	5.1521E−1 ± 1.53E−2	7.2757E−1 ± 3.80E−2	**7.3183E−1 ± 2.00E−2**
WFG7	6.1116E−1 ± 7.31E−3	**6.1254E−1 ± 4.73E−3**	8.2764E−1 ± 9.16E−3	**8.2897E−1 ± 1.17E−2**
WFG8	**3.7022E−1 ± 2.14E−2**	3.6682E−1 ± 2.72E−2	**5.9964E−1 ± 4.96E−2**	5.8005E−1 ± 3.96E−2
WFG9	4.9760E−1 ± 6.52E−2	**5.2891E−1 ± 4.38E−2**	6.8623E−1 ± 7.20E−2	**7.0326E−1 ± 4.68E−2**

The best result on each test instance is in bold

Table 9 HV values obtained by Two_Arch2 with accurate non-dominated sorting method and A-ENS on DTLZ1–DTLZ7 and WFG1–WFG9 with 5 and 10 objectives

Problems	$M = 5$		$M = 10$	
	T-ENS	A-ENS	T-ENS	A-ENS
DTLZ1	**8.6562E−1 ± 2.11E−2**	8.2886E−1 ± 2.08E−2	**8.7591E−1 ± 4.92E−2**	7.9605E−1 ± 4.67E−2
DTLZ2	**5.7334E−1 ± 5.88E−3**	5.6916E−1 ± 7.30E−3	4.7341E−1 ± 3.09E−2	**6.7170E−1 ± 2.89E−2**
DTLZ3	**3.0047E−1 ± 4.14E−2**	2.4114E−1 ± 3.13E−2	2.3976E−1 ± 1.27E−1	**2.5976E−1 ± 5.31E−2**
DTLZ4	**5.4961E−1 ± 1.05E−2**	5.4904E−1 ± 1.15E−2	5.4429E−1 ± 3.98E−2	**7.0587E−1 ± 1.46E−2**
DTLZ5	1.3372E−1 ± 7.98E−3	**1.3770E−1 ± 8.76E−3**	5.2870E−2 ± 1.55E−2	**8.0355E−2 ± 2.34E−2**
DTLZ6	1.1627E−1 ± 1.09E−2	**1.2888E−1 ± 1.81E−2**	9.8949E−3 ± 1.54E−2	**5.9620E−2 ± 1.26E−2**
DTLZ7	1.3042E−1 ± 1.65E−2	**1.4944E−1 ± 1.16E−2**	8.4290E−2 ± 2.32E−2	**1.0447E−1 ± 1.15E−2**
WFG1	9.8852E−1 ± 1.22E−3	**9.8861E−1 ± 1.48E−3**	9.9069E−1 ± 1.29E−3	**9.9436E−1 ± 8.20E−4**
WFG2	9.1682E−1 ± 9.08E−2	**9.5573E−1 ± 6.89E−2**	9.9005E−1 ± 3.14E−3	**9.9036E−1 ± 2.82E−3**
WFG3	**5.6637E−1 ± 1.03E−2**	5.6437E−1 ± 1.20E−2	5.6233E−1 ± 8.87E−3	**5.7551E−1 ± 1.61E−2**
WFG4	5.1069E−1 ± 1.00E−2	**5.1718E−1 ± 9.87E−3**	4.9015E−1 ± 1.85E−2	**5.1932E−1 ± 2.24E−2**
WFG5	4.9430E−1 ± 7.43E−3	**5.0298E−1 ± 6.10E−3**	4.7617E−1 ± 2.18E−2	**5.4869E−1 ± 1.80E−2**
WFG6	**4.8227E−1 ± 1.95E−2**	4.8073E−1 ± 1.32E−2	4.5144E−1 ± 2.37E−2	**5.3360E−1 ± 3.17E−2**
WFG7	5.5479E−1 ± 7.28E−3	**5.6390E−1 ± 7.92E−3**	5.2117E−1 ± 1.95E−2	**5.6443E−1 ± 2.51E−2**
WFG8	3.4962E−1 ± 1.44E−2	**3.6521E−1 ± 2.29E−2**	2.8372E−1 ± 3.96E−2	**3.4275E−1 ± 5.26E−2**
WFG9	**4.9517E−1 ± 7.85E−3**	4.9494E−1 ± 3.12E−2	4.5549E−1 ± 2.92E−2	**4.9367E−1 ± 2.83E−2**

The best result on each test instance is in bold

and WFG test problems with 5 and 10 objectives. It can also be found that the superiority of A-ENS over T-ENS in computational efficiency will be enhanced as the number of objectives increases to 10. The above results demonstrate the competitiveness of approximate non-dominated sorting in computational efficiency for handling many-objective optimization.

To evaluate the influence of A-ENS on performance of MOEAs, Tables 8, 9 present the hypervolume (HV) values obtained by KnEA and Two_Arch2 with A-ENS and accurate non-dominated sorting algorithm T-ENS on 5- and 10-objective DTLZ and WFG test problems. HV is a performance indicator to measure the quality of solution sets obtained by MOEAs in terms of both convergence and diversity [61]. The method for calculating HV value is the same to that adopted in [23]. The larger the value of HV, the better the solution set. From the tables, the following three results can be observed.

First, A-ENS can enhance the performance of both KnEA and Two_Arch2 in solving most 5- and 10-objective DTLZ and WFG test problems without considerably deterioration on all test instances under consideration. For the 32 test instances, A-ENS achieves better HV values than accurate non-dominated sorting on 19 test instances in KnEA, and 24 instances in Two_Arch2. The enhanced performance of KnEA and Two_Arch2 may show that the errors introduced by approximate non-dominated sorting are helpful for MOEAs to improve the performance in solving MaOPs.

Second, compared to the performance on 5-objective MaOPs, the effectiveness of A-ENS is significantly enhanced on 10-objective MaOPs for both KnEA and Two_Arch2. A-ENS obtains better HV values on 8 out of 16 test instances with 5 objectives in KnEA and 9 instances in Two_Arch2. The numbers of A-ENS outperforming accurate non-dominated sorting increase to 11 in KnEA and 15 in Two_Arch2 on 16 test instances with 10 objectives. Third, A-ENS is more helpful for Two_Arch2 than KnEA in improving their performance to solve MaOPs. This implies that the idea of approximate non-dominated sorting deserves further investigation by developing MOEAs well suited for approximate non-dominated sorting as reported in [31].

From the above empirical results, we can conclude that approximate non-dominated sorting A-ENS is a promising idea to perform non-dominated sorting for many-objective optimization, which cannot only improve the computational efficiency, but also enhance the performance in quality of solution set, when it is adopted in MOEAs to solve MaOPs.

Conclusion

In this paper, we have empirically analyzed the effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization. The effectiveness of non-dominated sorting is verified by considering two MOEAs, NSGA-II and KnEA, both of which adopted non-dominated sorting as an important component, to solve MOPs and MaOPs, respectively. Experimental results obtained by NSGA-II demonstrate that non-dominated sorting is very important for MOEAs to converge to the Pareto fronts when they are used to solve MOPs. For MaOPs, non-dominated sorting has been shown to be effective in MOEAs such as KnEA, especially for dealing with MaOPs with a large number of local Pareto fronts, despite that it suffers from the deterioration of effectiveness due to the dominance resistance phenomenon. Some enhanced variants of the Pareto dominance for many-objective optimization have also been briefly introduced.

The efficiency of non-dominated sorting is evaluated by comparing 8 state-of-the-art non-dominated sorting algorithms for evolutionary multi- and many-objective optimization. According to the experimental results, ENS-SS performs the best in efficiency for multi-objective optimization and T-ENS holds the best efficiency for many-objective optimization. The approximate non-dominated sorting algorithm A-ENS has also been empirically discussed and experimental results have indicated that approximate non-dominated sorting is a promising idea for many-objective optimization in terms of both efficiency and effectiveness.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61672033, 61502004, 61502001), and the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (Grant No. 61428302).

References

1. Gupta A, Mańdziuk J, Ong YS (2015) Evolutionary multitasking in bi-level optimization. Complex Intell Syst 1:83–95
2. Nguyen S, Mei Y, Zhang M (2017) Genetic programming for production scheduling: a survey with a unified framework. Complex Intell Syst 3:1–26
3. Ponsich A, Jaimes AL, Coello CAC (2013) A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. IEEE Trans Evol Comput 17(3):321–344
4. Zhang L, Pan H, Su Y, Zhang X, Niu Y (2017) A mixed representation-based multiobjective evolutionary algorithm for overlapping community detection. IEEE Trans Cybern 47(9):2703–2716
5. Zhang X, Duan F, Zhang L, Cheng F, Jin Y, Tang K (2017) Pattern recommendation in task-oriented applications: a multi-objective perspective. IEEE Comput Intell Mag 12:43–53
6. Cheng R, Li M, Tian Y, Zhang X, Yang S, Jin Y, Yao X (2017) A benchmark test suite for evolutionary many-objective optimization. Complex Intell Syst 3:67–81
7. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
8. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Proceedings of the 5th conference on evolutionary methods for design, optimization and control with applications to industrial problems, pp 95–100
9. Corne DW, Jerram NR, Knowles JD, Oates MJ (2001) PESA-II: region-based selection in evolutionary multi-objective optimiza-

tion. In: Proceedings of the 2001 conference on genetic and evolutionary computation, pp 283–290
10. Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Proceedings of the 8th international conference on parallel problem solving from nature, pp 832–842
11. Zhang Q, Li H (2007) MOEA/D: a multi-objective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
12. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Boston
13. Miettinen K (1999) Nonlinear multiobjective optimization. Springer, New York
14. Srinivas N, Deb K (1995) Multiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248
15. Kukkonen S, Lampinen J (2005) GDE3: the third evolution step of generalized differential evolution. In: Proceedings of the 2005 IEEE congress on evolutionary computation, vol 1, pp 443–450
16. Nebro AJ, Durillo JJ, Garcia-Nieto J, Coello CC, Luna F, Alba E (2009) SMPSO: a new PSO-based metaheuristic for multi-objective optimization. In: Proceedings of the 2009 IEEE symposium on computational intelligence in multi-criteria decision-making, pp 66–73
17. Cai X, Li Y, Fan Z, Zhang Q (2015) An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. IEEE Trans Evol Comput 19(4):508–523
18. Tian Y, Zhang X, Cheng R, Jin Y (2016) A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric. In: Proceedings of the 2016 IEEE congress on evolutionary computation, pp 5222–5229
19. Li B, Li J, Tang K, Yao X (2015) Many-objective evolutionary algorithms: a survey. ACM Comput Surv 48(1):13
20. Yang S, Li M, Liu X, Zheng J (2013) A grid-based evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 17(5):721–736
21. Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601
22. Zhang X, Tian Y, Jin Y (2014) A knee point driven evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 19(6):761–776
23. Zhang X, Tian Y, Cheng R, Jin Y (2017) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Trans Evol Comput (99):1–1. doi:10.1109/TEVC.2016.2600642
24. Li K, Deb K, Zhang Q, Kwong S (2015) An evolutionary many-objective optimization algorithm based on dominance and decomposition. IEEE Trans Evol Comput 19:694–716
25. Li M, Yang S, Liu X (2016) Pareto or non-Pareto: bi-criterion evolution in multiobjective optimization. IEEE Trans Evol Comput 20(5):645–665
26. Jensen MT (2003) Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. IEEE Trans Evol Comput 7(5):503–515
27. Deb K, Tiwari S (2005) Omni-optimizer: a procedure for single and multi-objective optimization. In: Proceedings of the 3rd international conference on evolutionary multi-criterion optimization, pp 47–61
28. Clymont KM, Keedwell E (2012) Deductive sort and climbing sort: new methods for non-dominated sorting. Evol Comput 20(1):1–26
29. Zhang X, Tian Y, Cheng R, Jin Y (2015) An efficient approach to non-dominated sorting for evolutionary multi-objective optimization. IEEE Trans Evol Comput 19(2):201–213
30. Wang H, Yao X (2014) Corner sort for Pareto-based many-objective optimization. IEEE Trans Cybern 44(1):92–102
31. Zhang X, Tian Y, Jin Y (2016) Approximate non-dominated sorting for evolutionary many-objective optimization. Inf Sci 369:14–33
32. Cheng R, Jin Y, Narukawa K, Sendhoff B (2015) A multiobjective evolutionary algorithm using Gaussian process based inverse modeling. IEEE Trans Evol Comput 19(6):838–856
33. Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multiobjective optimization test problems. In: Proceedings of the 2002 IEEE congress on evolutionary computation, pp 825–830
34. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
35. Ishibuchi H, Tsukamoto N, Nojima Y (2008) Evolutionary many-objective optimization: a short review. In: Proceedings of the 2008 IEEE congress on evolutionary computation, pp 2419–2426
36. Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multiobjective optimization. Evol Comput 10(3):263–282
37. Hernández-Díaz AG, Santana-Quintero LV, Coello CAC, Molina J (2007) Pareto-adaptive ϵ-dominance. Evol Comput 15(4):493–517
38. Batista LS, Campelo F, Guimarães FG, Ramírez JA (2011) Pareto cone ε-dominance: improving convergence and diversity in multiobjective evolutionary algorithms. In: Proceedings of the 2011 international conference on evolutionary multi-criterion optimization, pp 76–90
39. Sato H, Aguirre H, Tanaka K (2007) Controlling dominance area of solutions and its impact on the performance of MOEAs. In: Proceedings of the 2007 international conference on evolutionary multi-criterion optimization, pp 5–20
40. Sato H, Aguirre HE, Tanaka K (2010) Self-controlling dominance area of solutions in evolutionary many-objective optimization. In: SEAL, vol 8, Springer, New York, pp 455–465
41. Ikeda K, Kita H, Kobayashi S (2001) Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In: Proceedings of the 2001 IEEE congress on evolutionary computation, pp 957–962
42. Zhu C, Xu L, Goodman ED (2016) Generalization of Pareto-optimality for many-objective evolutionary optimization. IEEE Trans Evol Comput 20(2):299–315
43. Farina M, Amato P (2004) A fuzzy definition of "optimality" for many-criteria optimization problems. IEEE Trans Syst Man Cybern Part A Syst Hum 34(3):315–326
44. Zou X, Chen Y, Liu M, Kang L (2008) A new evolutionary algorithm for solving many-objective optimization problems. IEEE Trans Syst Man Cybern Part B 38(5):1402–1412
45. Wang G, Jiang H (2007) Fuzzy-dominance and its application in evolutionary many objective optimization. In: Proceedings of the 2007 international conference on computational intelligence and security workshops, pp 195–198
46. He Z, Yen GG, Zhang J (2014) Fuzzy-based Pareto optimality for many-objective evolutionary algorithms. IEEE Trans Evol Comput 18(2):269–285
47. Elarbi M, Bechikh S, Gupta A, Said LB, Ong YS (2017) A new decomposition-based NSGA-II for many-objective optimization. IEEE Trans Syst Man Cybern Syst PP(99):1–20. doi:10.1109/TSMC.2017.2654301
48. Yuan Y, Xu H, Wang B, Yao X (2016) A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 20(1):16–37
49. Bader J, Zitzler E (2011) HypE: an algorithm for fast hypervolume-based many-objective optimization. Evol Comput 19(1):45–76
50. Li M, Yang S, Liu X (2014) Shift-based density estimation for Pareto-based algorithms in many-objective optimization. IEEE Trans Evol Comput 18(3):348–365

51. Zhang X, Tian Y, Cheng R, Jin Y (2016) Empirical analysis of a tree-based efficient non-dominated sorting approach for many-objective optimization. In: Proceedings of the 2016 IEEE symposium series on computational intelligence, pp 1–8
52. Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors. J ACM 22:469–476
53. Fang H, Wang Q, Tu Y, Horstemeyer MF (2008) An efficient non-dominated sorting method for evolutionary algorithms. Evol Comput 16(3):355–384
54. Fortin FA, Grenier S, Parizeau M (2013) Generalizing the improved run-time complexity algorithm for non-dominated sorting. In: Proceedings of the 15th annual conference on genetic and evolutionary computation, pp 615–622
55. Drozdík M, Akimoto Y, Aguirre H, Tanaka K (2015) Computational cost reduction of non-dominated sorting using M-front. IEEE Trans Evol Comput 19(5):659–678
56. Veldhuizen DAV, Lamont GB (1998) Multiobjective evolutionary algorithm research: a history and analysis. Technical report, Department of Electrical and Computer Engineering. Graduate School of Engineering, Air Force Inst Technol, Wright Patterson, Tech. Rep. TR-98-03
57. Zhou A, Jin Y, Zhang Q, Sendhoff B, Tsang E (2006) Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: Proceedings of the 2006 IEEE congress on evolutionary computation, pp 892–899
58. Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multiobjective optimization. Springer, New York
59. Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evol Comput 10(5):477–506
60. Wang H, Jiao L, Yao X (2015) Two_arch2: an improved two-archive algorithm for many-objective optimization. IEEE Trans Evol Comput 19(4):524–541
61. While L, Hingston P, Barone L, Huband S (2006) A faster algorithm for calculating hypervolume. IEEE Trans Evol Comput 10(1):29–38

Publisher's Note