Empirical Analysis of A Tree-based Efficient Non-dominated Sorting Approach for Many-Objective Optimization

Xingyi Zhang[1], Ye Tian[1], Ran Cheng[2] and Yaochu Jin[3]
[1]Institute of Bio-inspired Intelligence and Mining Knowledge,
School of Computer Science and Technology, Anhui University, Hefei 230039, China
[2]School of Computer Science, University of Birmingham, Birmingham, B15 2TT, United Kingdom
[3]Department of Computer Science, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom
Email: xyzhanghust@gmail.com; field910921@gmail.com; ranchengcn@gmail.com; yaochu.jin@surrey.ac.uk

***Abstract*—Non-dominated sorting has been widely adopted in evolutionary multi-objective optimization. Many approaches to non-dominated sorting have been proposed to improve its computational efficiency, but unfortunately, most of them still suffer from high computational cost, especially when the number of objectives becomes large. A tree-based efficient non-dominated sorting approach, termed T-ENS, has been recently developed by us for many-objective optimization, where a tree structure is adopted to represent solutions, such that the non-dominance relationship between solutions can be easily inferred from the position of the solutions in the tree, thereby considerably reducing the number of comparisons between solutions belonging to the same non-dominated front. To validate the computational efficiency of T-ENS, this paper provides a detailed empirical analysis by comparing T-ENS with the state-of-the-art approaches, in particular when the number of objectives is larger than three and the population size becomes large. Empirical results indicate that the T-ENS is well suited for evolutionary many-objective optimization and large-scale multi-objective optimization, where either the number of objectives or the population size is large.**

I. Introduction

Pareto-based approach is one main mechanism for environmental selection in evolutionary multi-objective optimization algorithms (MOEAs), in which non-dominated sorting has been considered as the most effective technique. Among many others, NSGA-II [1] and SPEA2 [2] are two representative MOEAs for solving multi-objective optimization problems (MOPs) that adopt non-dominated sorting for environmental selection. Recently, increasing efforts have been devoted to MOPs with more than three objectives, often known as many-objective optimization problems (MaOPs), where non-dominated sorting has also been adopted as the first criterion for environmental selection, often being combined with a second convergence-related criterion to increase the selection pressure. Examples of MOEAs for MaOPs using non-dominated sorting as the first selection criterion include HypE [3], PICEA-g [4], GrEA [5], NSGA-III [6], RVEA [7] and KnEA [8], among many others.

However, most non-dominated sorting approaches suffer from high computational cost, especially when a very large population is used (for large-scale multi-objective optimization) or the number of objectives is large. Typically, it is also necessary to use a larger population for solving MaOPs than MOPs. Since non-dominated sorting needs to be performed at each generation, Pareto-based MOEAs will become highly time-consuming for solving MaOPs or large-scale MOPs.

Due to the crucial role of non-dominated sorting in the design of effective Pareto-based MOEAs, much research work has been dedicated to the improvement of computational efficiency of non-dominated sorting since the first non-dominated sorting algorithm was suggested in [9]. The non-dominated sorting method reported in [9] has a time complexity of $O(MN^3)$ and a space complexity of $O(N)$, where M is the number of objectives and N is the number of solutions to be sorted. Roughly speaking, existing improved non-dominated sorting algorithms can be divided into the following five categories.

The first category includes two non-dominated sorting approaches, known as fast non-dominated sort [1] and better non-dominated sort [10]. In these approaches, for each solution p in the population, the number of solutions that dominate solution p is countered, which is denoted by N_p. Meanwhile, all solutions that solution p dominates are recorded in a set denoted by S_p. Once this is complete, all solution whose $N_p = 0$ are assigned to the first non-dominated front. Then, for each solution q in set S_p of solution p in the first front, $N_q = N_q - 1$. Any solution q whose $N_q = 0$ is consequently assigned to the second non-dominated front. This process continues until all solutions in the population are assigned to a non-dominated front. Compared with the non-dominated sorting approach suggested in [9], the time complexity of the fast non-dominated sort has been significantly improved, achieving a complexity of $O(MN^2)$. The reduction of complexity lies in the fact that any two solutions need to be compared only once for calculating N_p and S_p. The better non-dominated sort is an improved version of the fast non-dominated sort, which takes advantage of the symmetry and transitivity properties of Pareto dominance to reduce the number of comparisons between solutions. However, the worst case time complexity of the better non-dominated sort is still the same as the fast non-dominated sort. Both sorting methods have a space complexity of $O(N^2)$.

The idea in the second category of non-dominated sorting approaches consists of following steps. First, find all non-dominated solutions in the population to be sorted and assign

them to the first front F_1. Second, remove the solutions assigned to F_1 and determine the non-dominated solutions in the remaining population, which are assigned to the second front F_2; then repeat this procedure until all solutions are assigned to a front. Most of existing non-dominated sorting approaches belong to this category, e.g., quick sort [11], non-dominated rank approach [12], sorting based algorithm [13], immune recognition based algorithm [14], arena's principle [15], deductive sort [16] and corner sort [17], including the first non-dominated sorting approach suggested in [9]. The main difference between the second category of non-dominated approaches lies in the strategy to reduce the number of comparisons between solutions in determining the solutions belonging to each non-dominated front. For instance, deductive sort reduces the number of comparisons between solutions by inferring some dominance relationships between solutions based on the recorded comparison results, while corner sort saves comparisons between solutions by ignoring the solutions dominated by a non-dominated solution selected from the corner solutions.

The third category of non-dominated sorting adopts the divide-and-conquer strategy developed by Kung *et al.* [18] for non-dominated sorting. The first attempt was made by Jensen, termed Jensen's sort [19]. Jensen's sort holds a time complexity of $O(N\ln^{M-1}N)$, which is computationally very efficient for MOPs with two or three objectives. Unfortunately, this sorting approach has two main weaknesses. The first weakness is that it does not work efficiently when there are more than three objectives. The second weakness of Jensen's sort is that it fails to work when two solutions have exactly the same value in any of their objectives. To address this issue, Fang *et al.* [20] suggested a new divide-and-conquer based non-dominated sorting approach. Fortin *et al.* [21] also developed an improved version of Jensen's sort to address the second weakness without increasing the time complexity and space complexity.

The fourth category of non-dominated sorting methods are developed for steady-state evolutionary algorithms, where only one solution in the parent population is updated at one time. The main idea is to determine the dominance relationship of the updated population by taking advantage of the known dominance relationship in the parent population. Two non-dominated sorting approaches belonging to this category have been reported, one termed efficient non-domination level update approach [22], and the other M-front [23].

The fifth category is to sort solutions in the population in an ascending order according to one of the objectives before the non-dominated sorting is performed. One unique property of pre-sorting the population is that in a sorted population, a solution can never be dominated by solutions ranked behind it. Consequently, solutions to be assigned to a front needs only to be compared with those that have already been assigned to a non-dominated front, which can spare a large number of unnecessary comparisons between solutions. Based on this idea, an efficient non-dominated sorting approach, called efficient non-dominated sort (ENS) was proposed in [24]. The ENS has been demonstrated a significant performance enhancement for MOPs with two or three-objective MOPs, its efficiency will decrease as the number of objectives increases despite that the superiority still exists compared with the state-of-the-art non-dominated sorting approaches. For addressing this issue, two ideas were recently developed on the basis of the ENS. The first one is to adopt the approximate non-dominated sorting and an approach using this idea termed A-ENS was suggested for many-objective optimization [25]. In A-ENS, the dominance relationship between two solutions is determined by a maximum of three objective comparisons, hence the complexity of A-ENS is independent of the number of objectives, while did not lead to the performance deterioration of MOEAs. The other idea is to reduce the large number of non-dominance comparisons between solutions in solving MaOPs and a tree based non-dominated sorting approach T-ENS was proposed using this idea [26]. In T-ENS, the information about the objectives to be used for identifying the non-dominance relationship is recorded in the nodes of the tree, by which a large number of non-dominance comparisons between solutions can be inferred. As a result, only a sub-set of the solutions, rather than all, that have been assigned to a non-dominated front needs to be compared, thus the T-ENS can save a large number of comparisons between solutions in the same front. Table I presents a summary of the state-of-the-art of non-dominated sorting methods, where the population to be sorted contains N solutions and M objectives.

In this work, we empirically verify the computational efficiency of T-ENS by comparing it with five state-of-the-art non-dominated sorting approaches. Two typical scenarios where non-dominated sorting is required are considered in empirical experiments. In the first scenario, we consider the non-dominated sorting to obtain a set of reference points uniformly distributed in the Pareto front, which are essential for some test problems, e.g., DTLZ7. The set of reference points is required in the calculation of some widely used performance metrics, such as GD [27] and IGD [28]. In the second scenario, we compare these non-dominated sorting approaches by embedding them in a recently developed MOEA specially designed for MaOPs, the knee point driven evolutionary algorithm (KnEA) [8] to test their computational performance in optimization. Empirical results on both scenarios confirm that the computational efficiency of T-ENS is much better than that of existing non-dominated sorting methods for MaOPs. In addition, the enhancement in computational efficiency of the T-ENS becomes increasingly significant as the number of objectives or the population size increases, which is particularly encouraging.

The rest of the paper is organized as follows. In Section II, we recall the details of the tree-based efficient non-dominated sorting approach, T-ENS. Simulation results are presented in Section III to empirically verify the computational efficiency of T-ENS. Finally, conclusions are drawn in Section IV.

II. The Non-dominated Sorting Approach T-ENS

Non-dominated sorting is a procedure that assigns candidate solutions in a population to different non-dominated fronts based on their dominance relationships. Without loss of generality, we assume that the individuals in the population P can be categorized into L non-dominated fronts, denoted as $F_1, F_2, \ldots, F_L$. According to the principle of non-dominated sorting, all non-dominated solutions in P are assigned to front F_1. Then solutions assigned to F_1 are temporarily removed from P and the non-dominated solutions in $P \backslash F_1$ are assigned

TABLE I. LISTING OF BEST CASE TIME COMPLEXITY, WORST CASE TIME COMPLEXITY AND SPACE COMPLEXITY OF EXISTING NON-DOMINATED SORTING APPROACHES.

Category	Approach	Authors	Year of Publication	Time Complexity		Space Complexity
				Best Case	Worst Case	
1	Fast Non-dominated Sort [1]	K. Deb, et al.	2002	$O(MN^2)$	$O(MN^2)$	$O(N^2)$
	Better Non-dominated Sort [10]	C. Shi, et al.	2005	$O(MN+N^2)$	$O(MN^2)$	$O(N^2)$
2	Non-dominated Sort [9]	N. Srinivas, et al.	1995	$O(MN^2)$	$O(MN^3)$	$O(N)$
	Quick Sort [11]	J. Zheng, et al.	2004	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$
	Non-dominated Rank Approach [12]	K. Deb, et al.	2006	$O(MN^2)$	$O(MN^3)$	$O(N)$
	Sorting Based Algorithm [13]	J. Du, et al.	2007	$O(MN\sqrt{N})$	$O(MN^2)$	$O(MN)$
	Immune Recognition Based Algorithm [14]	X. Zhou, et al.	2008	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$
	Arena's Principle [15]	S. Tang, et al.	2008	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$
	Deductive Sort [16]	K. McClymont, et al.	2012	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$
	Corner Sort [17]	H. Wang, et al.	2013	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$
3	Jensen's Sort [19]	M. T. Jensen	2003	$O(N\ln^{M-1}N)$	$O(N\ln^{M-1}N)$	$O(N)$
	Divide-and-Conquer based Non-dominated Sorting Algorithm [20]	H. Fang, et al.	2008	$O(MN\ln N)$	$O(MN^2)$	$O(N)$
	Generalized Jensen's Sort [21]	F.-A. Fortin, et al.	2013	$O(N\ln N)$	$O(MN^2)$	$O(N)$
4	Efficient Non-Domination Level Update Approach [22]†	K. Li, et al.	2014	$O(M)$	$O(MN\sqrt{N})$	$O(N)$
	M-front [23]	M. Drozdík, et al.	2014	$O(MN)$ or $O(MN\ln N)$ using K-d tree	$O(MN^2)$	$O(MN)$
5	Efficient Non-dominated Sort (termed ENS) [24]	X. Zhang, et al.	2014	$O(MN\ln N)$ for ENS-BS, $O(MN\sqrt{N})$ for ENS-SS	$O(MN^2)$	$O(1)$
	A-ENS [25]	X. Zhang, et al.	2016	$O(N\sqrt{N})$	$O(N^2)$	$O(N)$
	T-ENS [26]	X. Zhang, et al.	2016	$O(MN\ln N/\ln M)$	$O(MN^2)$	$O(MN)$

†Note that the time complexity listed here is the one that a new solution is added to a population whose non-domination level structure has been known, since the approach is designed for the MOEAs that the population should be updated whenever a new candidate solution is reproduced.

to front F_2. This procedure repeats until all solutions in P are assigned to a non-dominated front F_i, $1 \leq i \leq L$. Solutions in front F_i are considered to be better than those in front F_j for $j > i$.

In this section, we first briefly review the ENS approach, then present a detailed description of T-ENS.

A. A Summary of ENS

For a minimization problem, the ENS approach performs as follows. First, all solutions in the population are sorted in an ascending order according to the first objective. Solutions will be sorted according to the j-th objective in case they have the same value on each of the i-th objective, $1 \leq i < j$. Second, solutions in the sorted population are assigned to non-dominated fronts one by one, starting from the first solution to the last one by comparing the solution to be assigned with those that have been assigned to the fronts. A solution will be assigned to a front if this solution is not dominated with all solutions that have been assigned to this front. Otherwise, ENS will check whether it can be assigned to the next front. The procedure repeats until all solutions in the population are assigned to a front. Fig. 1 illustrates the sorting process of ENS with a population having four solutions.

In ENS, a solution to be assigned to a front only needs to be compared with solutions that have been assigned to the fronts. Therefore, ENS can avoid a large number of unnecessary comparisons between solutions, which makes the computational efficiency of ENS very competitive for populations with a small number of objectives. However, the efficiency of ENS will decrease as the number of objectives increases due to the large number of comparisons between solutions in the same front. The rapid degradation of the computational efficiency of ENS as the number of objectives increases can be attributed to the fact that in ENS a solution to be assigned to a front must be compared all solutions that have been assigned to the front before it can be assigned to the front. In dealing with optimization problems with a large number of objectives, the population of MOEAs only consists of one or two non-dominated fronts already in the early search stage.

B. The T-ENS Approach

In order to reduce the number of comparisons between solutions in the same fronts as much as possible, the T-ENS

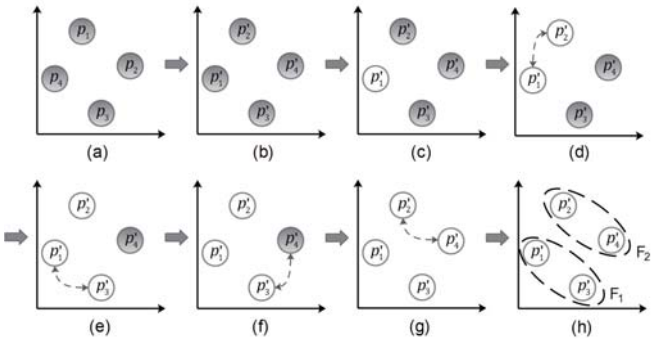

Fig. 1. An illustration of the main steps in the ENS framework. (a) A population consists of four individuals, p_1, p_2, p_3 and p_4; (b) The four individuals are sorted in an ascending order according to the first objective, resulting in p'_1, p'_2, p'_3 and p'_4; (c) The first solution p'_1 in the sorted population is assigned to the first front F_1; (d) The second solution p'_2 in the sorted population is compared with p'_1 and it is assigned to the second front F_2 due to the fact that it is dominated by p'_1; (e) The third solution p'_3 in the sorted population is compared with p'_1 and p'_3 is assigned to the first front F_1 since it is non-dominated with the only solution p'_1 that has been assigned to F_1; (f) The fourth solution p'_4 is compared with p'_3 in F_1 and it is found that it does not belong to F_1 since p'_4 is dominated by p'_3; (g) p'_4 is compared with p'_2 in the second front F_2 and p'_4 is assigned to F_2 since it is non-dominated with the solution p'_2 that has been assigned to F_2; (h) The non-dominated sorting is thus complete.

proposes to use a tree to represent the solutions in each non-dominated front. The benefit of adopting a tree structure, compared to the set based representation used in ENS, is that information about the objectives determining the non-dominance relationship between solutions can be recorded by the position of nodes in the tree in which the solutions are stored. In other words, if a population consists of L non-dominated fronts, T-ENS will construct L trees, each representing the non-dominated solutions that belong to a front. As a result, many non-dominance relationships between solutions can be inferred from those that have been assigned to the front (stored in the tree), and a solution to be assigned to a front only needs to be compared with some of them, provided that the solution to be assigned eventually belongs to this front. Therefore, compared to ENS, T-ENS can spare a large number of comparisons between solutions belonging to the same front, which can lead to significant enhancement of the computational efficiency of T-ENS over ENS, in particular for solving MaOPs. Algorithm 1 presents the main steps of T-ENS in pseudo code taken from [26].

To describe how the tree of each front is constructed in T-ENS, let us consider an illustrative example of a population consisting of the following six solutions for a 4-objective minimization problem: $(3, 5, 3, 2)$, $(4, 1, 3, 2)$, $(1, 3, 4, 2)$, $(5, 2, 4, 3)$, $(2, 4, 4, 1)$, $(6, 2, 4, 1)$, where each solution is represented by their four objective values in the form of (f_1, f_2, f_3, f_4). The non-dominated sorting procedure of T-ENS is performed as follows. First, as in ENS, the six solutions in the population are sorted according to the first objective value in an ascending order. For convenience, the sorted solutions in the population are denoted by $p_i(f_1, f_2, f_3, f_4)$, where $1 \leq i \leq 6$: $p_1(1, 3, 4, 2)$, $p_2(2, 4, 4, 1)$, $p_3(3, 5, 3, 2)$, $p_4(4, 1, 3, 2)$, $p_5(5, 2, 4, 3)$ and $p_6(6, 2, 4, 1)$. Once sorting is completed, T-ENS starts to construct the tree of the first front F_1 by checking solutions in the sorted population from the first solution p_1 to the last one p_6.

Algorithm 1: The main steps of T-ENS

Input: P (population), M (number of objectives)
Output: F (set of fronts, each front is represented by a tree)

```
Sort P in an ascending order according to the first objective;
F ← ∅;
k ← 0;
while not_empty(P) do
    k ← k + 1 ; /*start to construct a new tree*/
    for all the p ∈ P do
        objSeq[p] ← random permutation from 2 to M;
        update_tree(p, F[k], objSeq);
return F;
```

Algorithm 2: $update\_tree(p, tree, objSeq)$

Input: p (the solution to be checked), $tree$ (the tree to be checked), $objSeq$ (the permutation of objectives for each solution)
Output: -

```
if empty(tree) then
    tree ← p ;   /*p is used as the root of the tree*/
else if check_tree(p, tree, objSeq, true) then
    P ← P \ {p};
return;
```

The first solution p_1 definitely belongs to the first front F_1 and will naturally be placed in the root node of the tree for F_1, as shown in Fig. 2(a). All other solutions belonging to F_1 will be stored as the descendants of p_1. Note that in the figure, the black circle denotes a solution to be assigned, while gray circles denote solutions with which the solution in the black circle needs to be compared to check whether it can be assigned to F_1. Solutions in white circles do not need to be compared with in assigning the solution in the black circle.

Next, T-ENS will check whether the second solution p_2 can be assigned to F_1 by comparing it with solution p_1. Since solution p_2 has a smaller value than p_1 on the fourth objective, p_2 is not dominated by solution p_1 and hence, p_2 can be assigned to the first front F_1. In the tree structure, p_2 will be added to the third child node of the root due to the fact the fourth objective of p_2 is smaller than that of p_1, which provides the information showing the objective that helps determine the non-dominance relationship between the two solutions, as depicted in Fig. 2(b).

After p_2 is assigned, T-ENS checks if p_3 can be assigned to the same front of p_1 and p_2. To this end, T-ENS only needs to compare p_3 with the root solution p_1, since the non-dominance relationship between p_3 and p_2 can be inferred from the non-dominance relationship between p_3 and p_1. It is noticed that p_3 has a smaller value on the third objective than solution p_1, while p_2 has a larger value on this objective than p_1. This

Algorithm 3: $check\_tree(p, tree, objSeq, add\_pos)$

Input: p (the solution to be checked), $tree$ (the tree to be checked), $objSeq$ (the permutation of objectives for each solution), add_pos (indicates whether p can be added to a node of $tree$ as a ray)

Output: nd (indicates whether p is non-dominated with all solutions that have been assigned to the $tree$)

```
if empty(tree) then
    return true ;    /*tree is an empty tree*/
Find the minimal index m satisfying that
p[objSeq[tree.root][m]] <
tree.root[objSeq[tree.root][m]];
if m not found then
    return false ;    /*p is dominated by the
    solution at the root*/
else
    for i ← 1 to m do
        if check_tree(p, tree.branch[i], objSeq,
        i == m && add_pos)==false then
            return false ;   /*p is dominated by
            a solution in the branch of
            the tree*/
    if empty(tree.branch[m]) && add_pos then
        tree.branch[m] = p ;      /*add p to the
        branch of the tree*/
    return true;
```

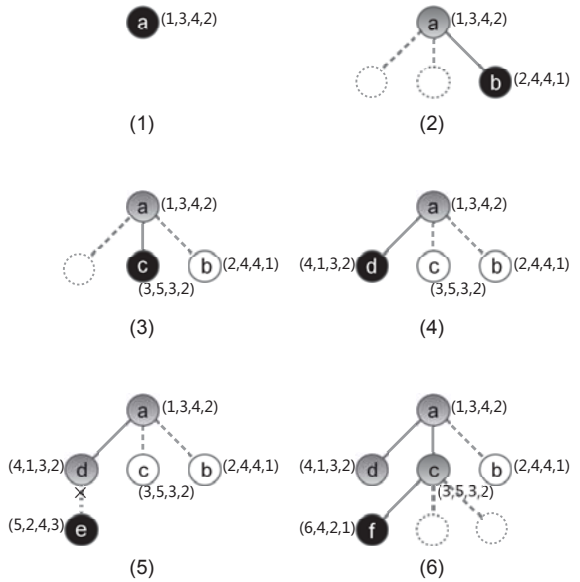

Fig. 2. An illustrative example showing the construction of the tree representing the solutions belonging to the first front for a population of a 4-objective minimization problem, where solutions in gray circles denote those with which a solution to be assigned (in the black circle) need to be compared to check whether it can be assigned to F_1. Solutions in white circles do not need to be compared in assigning the solution in the black circle.

means that solution p_2 will have a larger value than p_3 on the third objective if p_1 has a larger value than p_3 on that objective. Because the third solution p_3 is non-dominated with solution p_1, it can be inferred that solution p_3 is also non-dominated with p_2, thus p_3 can also be assigned to the first front F_1. To repeat, since p_3 belongs to the same front of p_1 and has a smaller value on the third objective than p_1, p_3 is placed as the second child (from the left) of the root. The tree after adding p_3 is shown in Fig. 2(c).

By simply comparing it with solution p_1, it can be found that solution p_4 also belongs to F_1 since the non-dominance relationship between p_4 and p_3 and between p_4 and p_2 can be inferred from the non-dominance relationship between p_4 and p_1, due to the fact that p_4 has a smaller value than p_1 on the second objective, whereas p_2 and p_3 have a larger value than p_1 on this objective. Therefore, solution p_4 is added to the first descendant node of the root, since p_4 has a smaller value than p_1 on the second objective. The tree of the first front after adding solution p_4 is shown in Fig. 2(d).

The next solution to check is p_5 and it is first compared with solution p_1. It is found that the second objective value of p_5 is smaller than that of p_1. Thus, solution p_5 is non-dominated with solution p_1 since the value of the first objective of p_1 is smaller than p_5. From this non-dominance relationship between p_1 and p_5, it can be inferred that p_5 is also non-dominated with solutions p_2 and p_3, since the value of the second objective of p_1 is smaller than that of p_2 and p_3. This implies that only p_4 needs to be compared to check whether p_5 can be assigned to front F_1. After comparison, it is found that solution p_4 dominates p_5, meaning that p_5 does not belong to F_1 and therefore the tree for F_1 remains unchanged, as illustrated in Fig. 2(e).

Finally, T-ENS checks if solution p_6 can be assigned to F_1. For this purpose, solution p_6 is compared with solution p_1 at first. The comparison shows that p_6 has a smaller value on the third objective than p_1. This indicates that solution p_6 is non-dominated with p_1 and the non-dominance relationship between p_6 and p_2 can be inferred from the non-dominance between p_1 and p_6. For determining whether p_6 can be assigned to F_1, we only need to compare it with p_3 and p_4. By comparing p_6 with p_3 and p_4, we can see that p_6 is also non-dominated with p_3 and p_4 and hence assign p_6 to F_1. Because it has a smaller value than p_1 on the third objective like p_3, p_6 is added as a child of p_3 and becomes a grandchild node of p_1. Moreover, by comparing with solution p_3, we find that p_6 has a smaller value than p_3 on the second objective, so p_6 is stored as the first child (from the left) of p_3, as shown in Fig. 2(f). The non-dominated sorting process is thus completed.

The above example illustrates the basic idea of the tree-based representation of solutions in the same front in T-ENS. Assume the population contains N M-objective solutions: $p_1(f_1^1, f_2^1, ..., f_M^1), p_2(f_1^2, f_2^2, ..., f_M^2), ..., p_N(f_1^N, f_2^N, ..., f_M^N)$, where f_j^i is the j-th objective value of the i-th solution, $1 \leq j \leq M$, and $1 \leq i \leq N$. Note that for minimization problems, we assume that the individuals have already been sorted according to an ascending order of the first objective. T-ENS aims to reduce the number of comparisons between solutions belonging to the same non-dominated front by storing these solutions in a specific position in the tree for the front. In performing non-dominated sorting, T-ENS adopts the first solution p_1 as the root of the tree for the

first non-dominated front F_1 and all other solutions in the population belonging to F_1 will be stored as the descendants of p_1. The position of solution p_i, $2 \leq i \leq N$ in the tree is determined by the minimum j ($j > 1$) satisfying that $f^i_{objSeq[1][j]} < f^1_{objSeq[1][j]}$, where $objSeq[1][j]$ is the j-th element of a permutation that is randomly generated from 2 to M for solution p_1. It is worth noting that in the example shown in Fig. 2, the permutation $objSeq[i]$ for solution i is fixed to $2, \ldots, M$ for the sake of simplicity. More precisely, a solution p_i with the $objSeq[1][j]$-th objective satisfying the above condition will be stored as the j-th child (from the left) of the root. If there is more than one solution that satisfies the above condition, the next solution will be stored as a child of p_i, i.e, a grandchild of p_1. This procedure repeats until all solutions in the population are checked. Algorithms 2 and 3 present a detailed description of updating the tree of a front in pseudo code taken from [26].

With the above tree constructed to represent solutions in a front, if a solution p to be assigned to the front is non-dominated with a solution q in the tree, where p has a smaller value than q on the $objSeq[q][j]$-th objective (p has a larger value than q on the first objective since q is ranked before p in the sorted population), then the non-dominance relationship between p and the solution stored as the k-th child of node q from the left and all descendants of this child can be inferred from the non-dominance relationship between q and p, since for each of these solutions s we have $f^p_{objSeq[q][j]} < f^q_{objSeq[q][j]} < f^s_{objSeq[q][j]}$, $j \in \{1, 2, \ldots, M-1\}$. Therefore, p does not need to be compared with these solutions, thus sparing a large number of comparisons between solutions.

In the above example, T-ENS performs eight comparisons between solutions in completing the non-dominated sorting, among which six comparisons are made between solutions in the same front and two between solutions in different fronts. However, 11 comparisons between solutions will be needed for ENS, ten of which are comparisons between solutions in the same front and one between solutions in different fronts. This is due to the fact that in ENS, a solution to be assigned to a front must be compared with all solutions that have been assigned to that front in case this solution belongs to this front. By contrast, in T-ENS, this solution only needs to be compared with some of the solutions that have been assigned to the front. This is made possible by the fact that the tree-based representation records the objective (except for the first objective) that helps identify the non-dominance relationship between two solutions assigned to the front. Therefore, T-ENS is an even more efficient non-dominated sorting approach than ENS in that it can avoid a large number of comparisons between solutions in the same front, which is essential for reducing computational cost when the number of objectives is large.

III. Simulation Experiments and Analysis

In this section, we verify the performance of T-ENS by empirically comparing it with three popular non-dominated sorting approaches, namely, deductive sort [16], corner sort [17] and ENS-SS [24]. The experiments are conducted on two different scenarios. In the first scenario (Scenario 1), the computational efficiency of the four compared non-dominated sorting approaches are compared in obtaining a set of reference points uniformly distributed in the Pareto front of DTLZ7. In the second scenario (Scenario 2), we compare the efficiency of these non-dominated sorting approaches by embedding them in KnEA for solving MaOP benchmark problem DTLZ2. In the experiments, the maximum number of generations in KnEA is set to 250, and the rest parameters are specified as recommended in [8]. In both scenarios, the number of comparisons between solutions and the amount of runtime are used to evaluate the computational efficiency of the four non-dominated sorting approaches under consideration. All simulations reported in this work are conducted on a PC with 2.3GHz Intel Core i7-3610QM CPU and the Windows 7 SP1 64 bit operating system.

A. Scenario 1

Fig. 3 presents the number of comparisons between solutions and runtime(s) of the four compared non-dominated sorting approaches for obtaining a set of reference points uniformly distributed of the Pareto front of DTLZ7, where the number of sampled points is fixed to 1000 and 10000, respectively. From this figure, we can find that the ENS-SS performs the best among the four compared non-dominated sorting approaches in obtaining a reference set of DTLZ7 with two objectives, in terms of both number of comparisons between solutions and runtime. The ENS-SS saves about one third runtime of deductive sort and three fifths runtime of corner sort and T-ENS for 2-objective DTLZ7. For DTLZ7 with more than three objectives, the ENS-SS also consumes less runtime than corner sort and deductive sort, despite that they take almost the same number of comparisons between solutions when the number of objectives increases to 5 due to its space complexity of $O(1)$.

Compared with ENS-SS, corner sort and deductive sort, the computational efficiency of T-ENS is much better on DTLZ7 with three or more objectives in terms of both number of comparisons between solutions and runtime. More encouragingly, the superiority of T-ENS will be greatly enhanced as the number of objectives increases. For DTLZ7 with three objectives, T-ENS needs roughly a half of runtime of ENS-SS and deductive sort, and one third of runtime of corner sort. However, as the number of objectives increases to 15, T-ENS takes only one sixth of runtime of ENS-SS, and one eighth of runtime of corner sort and deductive sort in obtaining a set of reference points uniformly distributed in Pareto front of DTLZ7. As can be seen from the figure, it seems that the superiority of T-ENS will also increase when more points are uniformly sampled from the Pareto front of DTLZ7. As the number of sampled points increases from 1000 to 10000, the runtime saved by T-ENS on 15-objective DTLZ7 will increase from seven eighths to nine tenths compared with ENS-SS, which is faster than corner sort and deductive sort on 15-objective DTLZ7.

Therefore, we can conclude that the computational efficiency of T-ENS is much higher than the three compared approaches, ENS-SS, corner sort and deductive sort, in obtaining sets of reference points uniformly distributed in Pareto front of DTLZ7 with three or more objectives.

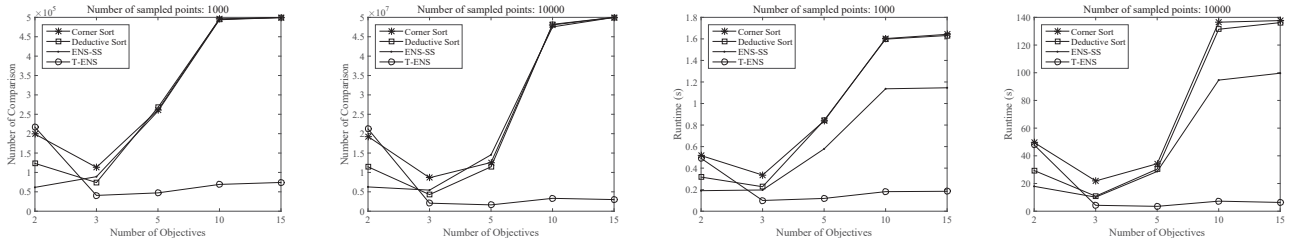

Fig. 3. Number of comparisons between solutions and runtime(s) of the four compared non-dominated sorting approaches for obtaining a set of reference points uniformly distributed in the Pareto front of DTLZ7 under different numbers of sampled points.

TABLE II. MEAN AND STANDARD DEVIATION OF NUMBERS OF COMPARISONS OF FOUR NON-DOMINATED SORTING APPROACHES WHEN THEY ARE EMBEDDED INTO KNEA TO SOLVE DTLZ2, AVERAGED OVER 20 RUNS. BEST PERFORMANCE IS SHOWN IN BOLD.

Population size	Obj.	Corner Sort	Deductive Sort	ENS-SS	T-ENS
100	2	3.5471e+6(2.98e+4)	3.4731e+6(2.54e+4)	**7.5093e+4(4.93e+2)**	3.6786e+6(2.48e+4)
	3	4.3370e+6(2.31e+4)	4.3772e+6(1.90e+4)	4.0933e+6(2.59e+4)	**1.0199e+6(3.34e+4)**
	5	4.6906e+6(8.56e+3)	4.7169e+6(7.44e+3)	4.6188e+6(9.27e+3)	**1.0011e+6(1.35e+4)**
	10	4.8537e+6(7.44e+4)	4.8686e+6(7.77e+4)	4.8582e+6(7.75e+4)	**1.3597e+6(2.80e+4)**
	15	4.9044e+6(2.47e+4)	4.9164e+6(2.51e+4)	4.9128e+6(2.34e+4)	**1.6145e+6(6.54e+4)**
500	2	8.5536e+7(1.26e+5)	7.7998e+7(1.67e+5)	**5.0924e+5(2.56e+3)**	8.7268e+7(1.69e+5)
	3	1.0684e+8(6.04e+4)	1.0736e+8(4.99e+4)	9.9915e+7(3.05e+4)	**1.4909e+7(2.37e+5)**
	5	1.1769e+8(1.55e+6)	1.1849e+8(1.48e+6)	1.1609e+8(1.69e+6)	**1.1243e+7(9.16e+4)**
	10	1.1165e+8(3.92e+6)	1.1174e+8(4.13e+6)	1.1367e+8(2.88e+6)	**1.3957e+7(3.41e+5)**
	15	1.2327e+8(3.09e+5)	1.2370e+8(2.16e+5)	1.2351e+8(2.90e+5)	**1.6126e+7(1.90e+5)**

TABLE III. MEAN AND STANDARD DEVIATION OF RUNTIME(S) OF FOUR NON-DOMINATED SORTING APPROACHES WHEN THEY ARE EMBEDDED INTO KNEA TO SOLVE DTLZ2, AVERAGED OVER 20 RUNS. BEST PERFORMANCE IS SHOWN IN BOLD.

Population size	Obj.	Corner Sort	Deductive Sort	ENS-SS	T-ENS	The rest operations in KnEA
100	2	1.5844e+1(3.86e-1)	1.5151e+1(3.26e-1)	**5.9564e-1(1.99e-2)**	1.4899e+1(3.78e-1)	5.3194e+0(1.54e-1)
	3	2.1231e+1(1.52e-1)	2.1160e+1(2.34e-1)	1.4360e+1(1.11e-1)	**4.6993e+0(1.41e-1)**	7.0481e+0(6.37e-1)
	5	2.4544e+1(3.01e-1)	2.4474e+1(5.33e-1)	1.6687e+1(1.82e-1)	**4.6854e+0(1.39e-1)**	7.2942e+0(3.24e-1)
	10	2.8917e+1(6.28e-1)	2.8729e+1(6.47e-1)	1.8182e+1(2.37e-1)	**6.0363e+0(3.41e-2)**	8.2037e+0(2.43e-1)
	15	3.8776e+1(6.65e+0)	3.8706e+1(6.29e+0)	2.2062e+1(2.81e+0)	**8.1395e+0(1.20e+0)**	8.7830e+0(9.02e-1)
500	2	3.4041e+2(1.30e+1)	3.0860e+2(1.17e+1)	**3.1573e+0(1.83e-1)**	3.1897e+2(1.41e+1)	9.8847e+1(5.60e-1)
	3	4.8867e+2(1.30e+1)	4.9248e+2(1.44e+1)	3.2974e+2(1.14e+1)	**6.0928e+1(1.75e+0)**	7.7333e+1(8.99e+0)
	5	5.3601e+2(2.49e+1)	5.4405e+2(2.46e+1)	3.6632e+2(1.84e+1)	**4.4031e+1(2.10e+0)**	1.1666e+2(3.18e+0)
	10	5.7568e+2(6.20e+1)	5.8237e+2(6.34e+1)	3.7746e+2(3.69e+1)	**5.2334e+1(3.27e+0)**	1.2749e+2(1.11e+1)
	15	8.7768e+2(6.77e+0)	1.0944e+3(1.08e+1)	5.3557e+2(2.40e+0)	**7.2145e+1(7.24e-1)**	9.3225e+1(1.81e+0)

B. Scenario 2

Tables II and III list the mean and standard deviation of number of comparisons and runtimes of four non-dominated sorting approaches when they are embedded into KnEA to solve DTLZ2 with a population size of 100 and 500 averaging over 20 runs, where the best performance is highlighted in bold. The runtime consumed by the rest operations in KnEA for solving DTLZ2 is also presented in Table III. From the tables, the following three observations can be made. First, compared to the rest operations in KnEA, non-dominated sorting is computationally highly time-consuming. Except for the proposed T-ENS, all other compared non-dominated sorting approaches take more than a half of the total runtime needed in KnEA.

Second, ENS-SS performs much better than corner sort, deductive sort and T-ENS on DTLZ2 with two objectives, whereas T-ENS performs the best on all DTLZ2 test instances with more than three objectives. On DTLZ2 with 5, 10 and 15 objectives, T-ENS consumes roughly two fifths of the runtime of ENS-SS and one fifth of that of deductive sort and corner sort respectively, when the population size of KnEA is 100. Note that the computational efficiency of T-ENS is also better than that of the compared algorithms on DTLZ2 with three objectives, since the number of fronts decreases to very small as the number of generations increases. Therefore, T-ENS is more suited for performing non-dominated sorting for Pareto-based MOEAs to solve MOPs with a large number of objectives.

Finally, T-ENS saves more runtime than the three compared non-dominated sorting approaches when a larger population size is used in KnEA. On DTLZ2 with 5, 10 and 15 objectives, T-ENS only needs about three fifteenths of the runtime of ENS-

SS and one tenth of corner sort and deductive sort, respectively, when the population size of KnEA is increased to 500 from 100.

From the above empirical results, we can conclude that T-ENS is a very competitive and promising non-dominated sorting algorithm for MOEAs to solve MaOPs. The larger the population size, the more superior T-ENS is compared to the state-of-the-art non-dominated sorting algorithms. This is particularly encouraging since a larger population size is often needed in MOEAs in dealing with MaOPs.

IV. Conclusions and Remarks

In this paper, we have empirically verified the computational efficiency of an efficient non-dominated sorting approach, termed T-ENS. The T-ENS has been recently developed on the basis of the ENS for non-dominated sorting in Pareto-based MOEAs to solve MaOPs [26]. Experimental results have clearly demonstrated that, compared with the state-of-the-art non-dominated sorting approaches, the ENS holds the best computational efficiency for 2-objective MOPs, while the T-ENS performs the best for MOPs with three or more objectives. Empirical results have also confirmed an encouraging performance of T-ENS that the superiority of T-ENS will be enhanced as the number of objectives and the size of population increases.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (Grant No. 61272152, 61502004, 61502001), the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (Grant No. 61428302) and EPSRC Grant (No. EP/K001523/1).

References

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

[2] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Fifth Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2001, pp. 95–100.

[3] J. Bader and E. Zitzler, "HypE: an algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

[4] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired co-evolutionary algorithms for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 474–494, 2013.

[5] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, 2013.

[6] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[7] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2016 (accepted).

[8] X. Zhang, Y. Tian, and Y. Jin, "A knee point driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation,*, dol.10.1109/TEVC.2014.2378512, 2014.

[9] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[10] C. Shi, M. Chen, and Z. Shi, "A fast nondominated sorting algorithm," in *Proceedings of 2005 International Conference on Neural Networks and Brain*, 2005, pp. 1605–1610.

[11] J. Zheng, C. X. Ling, Z. Shi, and Y. Xie, "Some discussions about MOGAs: individual relations, non-dominated set, and application on automatic negotiation," in *Proceedings of 2004 IEEE Congress on Evolutionary Computation*, 2004, pp. 706–712.

[12] K. Deb and S. Tiwari, "Omni-optimizer: a procedure for single and multi-objective optimization," in *Proceedings of the Third international conference on Evolutionary Multi-Criterion Optimization*, 2005, pp. 47–61.

[13] J. Du, Z. Cai, and Y. Chen, "A sorting based algorithm for finding non-dominated set in multi-objective optimization," in *Proceedinds of the Third International Conference on Natural Computation*, 2007, pp. 436–440.

[14] X. Zhou, J. Shen, and J. Shen, "An immune recognition based algorithm for finding non-dominated set in multi-objective optimization," in *Proceedings of 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008, pp. 305–310.

[15] S. Tang, Z. Cai, and J. Zheng, "A fast method of constructing the non-dominated set: arena's principle," in *Proceedinds of the Fourth International Conference on Natural Computation*, 2008, pp. 391–395.

[16] K. M. Clymont and E. Keedwell, "Deductive sort and climbing sort: new methods for non-dominated sorting," *Evolutionary Computation*, vol. 20, no. 1, pp. 1–26, 2012.

[17] H. Wang and X. Yao, "Corner sort for pareto-based many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 92–102, 2014.

[18] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM*, vol. 22, pp. 469–476, 1975.

[19] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, 2003.

[20] H. Fang, Q. Wang, Y. Tu, and M. F. Horstemeyer, "An efficient non-dominated sorting method for evolutionary algorithms," *Evolutionary Computation*, vol. 16, no. 3, pp. 355–384, 2008.

[21] F.-A. Fortin, S. Grenier, and M. Parizeau, "Generlizing the improved run-time complexity algorithm for non-dominated sorting," in *Proceedings of the fifteenth annual conference on Genetic and evolutionary computation*, 2013, pp. 615–622.

[22] K. Li, K. Deb, Q. Zhang, and S. Kwong, "Efficient non-domination level update approach for steady-state evolutionary multiobjective optimization," Department of Electtrical and Computer Engineering, Michigan State University, East Lansing, USA, Tech. Rep. COIN Report Number 2014014, 2014.

[23] M. Drozdík, Y. Akimoto, H. Aguirre, and K. Tanaka, "Computational cost reduction of non-dominated sorting using m-front," *IEEE Transactions on Evolutionary Computation*, 2014, dol. 10.1109/TEVC.2014.2366498.

[24] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to non-dominated sorting for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 201–213, 2015.

[25] X. Zhang, Y. Tian, and Y. Jin, "Approximate non-dominated sorting for evolutionary many-objective optimization," *Information Sciences*, vol. 369, pp. 14–33, 2016.

[26] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2016, in press.

[27] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Department of Electrical and Computer Engineering. Graduate School of Engineering, Air Force Inst Technol, Wright Patterson, Tech. Rep. TR-98-03, Tech. Rep., 1998.

[28] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. D. Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.