This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

IEEE TRANSACTIONS ON CYBERNETICS   1

Efficient Nondomination Level Update Method for Steady-State Evolutionary Multiobjective Optimization

Ke Li, Kalyanmoy Deb, *Fellow, IEEE*, Qingfu Zhang, *Senior Member, IEEE*, and Qiang Zhang

Abstract—Nondominated sorting (NDS), which divides a population into several nondomination levels (NDLs), is a basic step in many evolutionary multiobjective optimization (EMO) algorithms. It has been widely studied in a generational evolution model, where the environmental selection is performed after generating a whole population of offspring. However, in a steady-state evolution model, where a population is updated right after the generation of a new candidate, the NDS can be extremely time consuming. This is especially severe when the number of objectives and population size become large. In this paper, we propose an efficient NDL update method to reduce the cost for maintaining the NDL structure in steady-state EMO. Instead of performing the NDS from scratch, our method only updates the NDLs of a limited number of solutions by extracting the knowledge from the current NDL structure. Notice that our NDL update method is performed twice at each iteration. One is after the reproduction, the other is after the environmental selection. Extensive experiments fully demonstrate that, comparing to the other five state-of-the-art NDS methods, our proposed method avoids a significant amount of unnecessary comparisons, not only in the synthetic data sets, but also in some real optimization scenarios. Last but not least, we find that our proposed method is also useful for the generational evolution model.

Index Terms—Computational complexity, nondominated sorting (NDS), nondomination level (NDL), Pareto dominance, steady-state evolutionary multiobjective optimization (EMO).

K. Li was with the CERCIA, School of Computer Science, University of Birmingham, Birmingham, B15 2TT, U.K. He is now with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, U.K. (e-mail: keli.genius@gmail.com).

K. Deb is with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: kdeb@egr.msu.edu).

Q. Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qingfu.zhang@cityu.edu.hk).

Q. Zhang is with the Sapienza University of Rome, 00185 Rome, Italy (e-mail: csqzhang@gmail.com).

I. INTRODUCTION

A MULTIOBJECTIVE optimization problem can be stated as follows:

$$\text{minimize } \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T$$
$$\text{subject to } \mathbf{x} \in \Omega \qquad (1)$$

where $\Omega = \prod_{i=1}^{n} [a_i, b_i] \subseteq \mathbb{R}^n$ is the decision (variable) space, $\mathbf{x} = (x_1, \ldots, x_n)^T \in \Omega$ is a candidate solution. $\mathbf{F} : \Omega \rightarrow \mathbb{R}^m$ constitutes m conflicting objective functions, and $\mathbb{R}^m$ is called the objective space. A solution $\mathbf{x}^1$ is said to Pareto dominate another one $\mathbf{x}^2$ (denoted as $\mathbf{x}^1 \preceq \mathbf{x}^2$) if it has at least one better objective while not being worse in any other objective.

Nondominated sorting (NDS) is a procedure that divides a population of solutions into several nondomination levels (NDLs) according to their dominance relationships. It gives a relative quality of solutions, belonging to a specific NDL, with respect to the others. The NDS is a basic step in the evolutionary multiobjective optimization (EMO), it becomes time-consuming with the increase of the number of objectives and population size. The first NDS algorithm was proposed in [1]. Its computational complexity is $\mathcal{O}(mN^3)$, where N is the population size. Later, the time-consuming problem of the NDS was recognized and addressed by Deb *et al.* [2]. They developed the fast NDS (FNDS) method which avoids some unnecessary dominance comparisons by taking advantages of the existing comparison results. Its computational complexity is reduced to $\mathcal{O}(mN^2)$. Inspired by the divide-and-conquer idea suggested in [3], Jensen [4] proposed an NDS method with a computational complexity of $\mathcal{O}(N \log^{m-1} N)$, a significant speedup and reduction. However, this method fails to deal with the situation when two solutions share the same value for a certain objective. By inferring dominance relationship based on the transitivity property of Pareto dominance and previous comparisons, McClymont and Keedwell [5] suggested two methods, called climbing sort and deductive sort (DS), to reduce the computational cost of the NDS. Although these two methods hold the same worst-case complexity of $\mathcal{O}(mN^2)$ as the FNDS, empirical studies showed that both of them outperform the FNDS in terms of CPU time and number of dominance comparisons. However, these two methods are designed specifically for populations, where the dominance relationships between solutions are relatively common, which unfortunately does not hold for many-objective problems with more than three objectives. In order to save the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

2   IEEE TRANSACTIONS ON CYBERNETICS

number of objective comparisons in many-objective scenarios, Wang and Yao [6] proposed a corner sort (CS) method. Its basic idea is to use the nondominated solutions to ignore the solutions that they dominate. Recently, Zhang *et al.* [7] developed a computationally efficient NDS method, where a solution only needs to compare with those sorted ones when it is going to be added to an NDL.

According to the selection scheme, the existing EMO has two evolution models: one is the generational evolution model and the other is the steady-state evolution model [8]. The major difference between them is the moment to perform the environmental selection. In the prior case, a population of offspring solutions are generated before competing with their parents; while in the latter case, the parent population is updated once a new candidate solution has been generated. Since the population can be updated immediately before generating a whole population of offspring, the elite information can be timely utilized. This characteristic can make a steady-state EMO algorithm be computationally faster for approaching the Pareto-optimal front than its generational counterpart on some problems. However, this "first come first serve" mechanism also has the risk of being trapped in local optima. In the EMO literature, there exists many algorithms based on the steady-state evolution model (e.g., [9]–[16]). In some recent studies (e.g., [8], [17], and [18]), the steady-state EMO algorithm has shown better performance, in terms of convergence and diversity, than its generational counterparts on some problems. To our best knowledge, most, if not all, studies on the NDS are discussed in the context of a generational evolution model, whereas few have considered the situation for a steady-state evolution model yet. Buzdalov *et al.* [19] presented an incremental NDS for the steady-state EMO. But unfortunately, this method can only work for the 2-D case.

In fact, the NDL structure of the parent population is already known before generating a new candidate solution. The incorporation of a new solution usually does not shake the entire NDL structure. On the contrary, only a limited number of solutions in the parent population need to change their NDLs. Therefore, it is unnecessary to perform the NDS from scratch each time. Moreover, the solution, which has to change its NDL, only need to move forward or backward one NDL. Bearing these properties in mind, this paper proposes an efficient NDL update (ENLU) method to reduce the cost for maintaining the NDL structure in the steady-state EMO. By using the ENLU method, a steady-state EMO algorithm only needs to perform the NDS once at the beginning, and it just updates the NDL structure thereafter. More specifically, after the reproduction, the ENLU method locates the NDL to which the new candidate belongs. Afterwards, it recursively finds the solutions that need to change their NDLs and move them backward to their next NDLs. Analogously, after the environmental selection, the ENLU method recursively finds those solutions that need to change their NDLs and move them forward to their prior NDLs. The time complexity of ENLU method is $\mathcal{O}(m)$ in the best case and $\mathcal{O}(mN^2)$ in the worst case. Although the ENLU method holds the same worst-case complexity as the FNDS method, extensive experiments demonstrate that it avoids a significant amount of unnecessary comparisons

---

Algorithm 1: Steady-State NSGA-II

Input: algorithm parameters
Output: population P

1 Initialize a population $P \leftarrow \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$;
2 **while** *termination criterion is not met* **do**
3   Mating selection and generate an offspring $\mathbf{x}^c$;
4   Use NDS to divide $P' \leftarrow P \bigcup \{\mathbf{x}^c\}$ into several NDLs, i.e., $F_1, \cdots, F_l$;
5   Identify the worst solution $\mathbf{x}' \in F_l$ and set $P \leftarrow P' \setminus \{\mathbf{x}'\}$;
6 **return** P;

---

in practice. Furthermore, we find that the ENLU method is also useful for the generational evolution model.

In the rest of this paper, we first discuss the motivations of this paper in Section II. Then, the implementation details of our proposed ENLU method are described step by step in Section III. Afterwards, its computational complexity is theoretically analyzed in Section IV. Next, Section V empirically investigates the performance of ENLU method on several synthetic data sets and real optimization scenarios. Finally, Section VI concludes this paper and provides some future directions.

II. MOTIVATIONS

In order to understand the basic principles of the steady-state evolution model, Algorithm 1 presents the pseudo-code of a steady-state version of the classic elitist NDS genetic algorithm (NSGA-II) [8]. At the beginning, a population P is initialized via a uniform sampling over the decision space (line 1 in Algorithm 1). During the main while loop, P is updated as soon as the generation of a new candidate solution $\mathbf{x}^c$. The environmental selection involves two steps. One is using the NDS to divide the hybrid population P', a combination of P and $\mathbf{x}^c$, into l ($1 \leq l \leq |P'|$) NDLs, i.e., $F_1, \ldots, F_l$ (line 4 in Algorithm 1). More specifically, all nondominated solutions are at first assigned to F_1. Afterwards, solutions assigned to F_1 are temporarily removed from P' and the non-dominated solutions in $P' \setminus F_1$ are assigned to F_2, so on and so forth. Note that each solution in F_i is either nondominated with or dominated by at least one solution in F_j, where $i > j$ and $i, j \in \{1, \ldots, l\}$. After the NDS, we eliminate the worst solution $\mathbf{x}'$ at the last NDL F_l from P' to form a new P for the next iteration (line 5 in Algorithm 1).

Since the NDS requires pair-wise dominance comparisons among solutions, it can be a very time-consuming part in an EMO algorithm. To illustrate this problem, we perform two simple experiments by using the steady-state NSGA-II on several DTLZ2 test instances [20]. In the first experiment, the population size is set to 100 as a constant, while the number of objectives grows from 2 to 20 with a step size 1. For the second experiment, the number of objectives is fixed to 5, while the population size increases from 100 to 2000 with a step size 100. The number of generations is set as 1000 for all cases. From Fig. 1, we clearly see that the NDS indeed consumes a
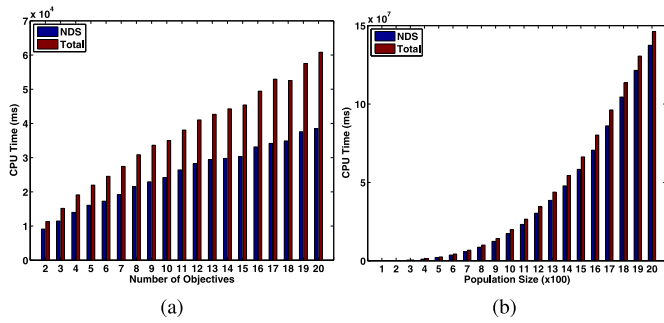
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LI *et al.*: ENLU METHOD FOR STEADY-STATE EMO

3

Fig. 1. Comparisons of CPU time (millisecond) cost by the NDS and the steady-state NSGA-II.

Fig. 3. $\mathbf{x}^4$, $\mathbf{x}^6$, and $\mathbf{x}^7$ need to change their NDLs when $\mathbf{x}^c$ is added and eliminated.
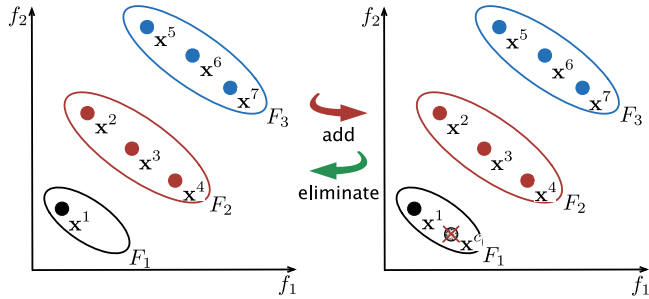

Fig. 2. NDL structure keeps unchanged when $\mathbf{x}^c$ is added and eliminated.

dominating amount of CPU time in the steady-state NSGA-II. Furthermore, the CPU time cost by the NDS increase with the number of objectives and the population size. One may argue that this ratio will changes in a computationally expensive optimization scenario, where the function evaluation is very time-consuming. Nevertheless, it is of significant importance in practice to reduce the cost of the NDS (or in other words, maintaining the NDL structure), especially for a large number of objectives and population size.

To this end, an idea naturally comes out: is it really necessary to perform the NDS from scratch, each time, during the environmental selection of the steady-state evolution model? Let us consider a simple example presented in Fig. 2, where there are three NDLs, i.e., $F_1 = \{\mathbf{x}^1\}$, $F_2 = \{\mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4\}$, and $F_3 = \{\mathbf{x}^5, \mathbf{x}^6, \mathbf{x}^7\}$. If a new candidate solution, say $\mathbf{x}^c$, comes in, none of these seven solutions need to change their NDLs and we only need to insert $\mathbf{x}^c$ into F_1. As for the other example shown in Fig. 3, $\mathbf{x}^4$, $\mathbf{x}^6$, and $\mathbf{x}^7$ need to move themselves backward to their next NDLs if the new candidate solution $\mathbf{x}^c$ comes in. Analogously, the NDL structure might also change after eliminating a solution by the environmental selection. Let us consider the same examples shown in Figs. 2 and 3 in an opposite direction. For simplicity, we assume that just $\mathbf{x}^c$ is eliminated after the environmental selection. For the example presented in Fig. 2, none of the remaining solutions need to change their NDLs, while for the example shown in Fig. 3, $\mathbf{x}^4$, $\mathbf{x}^6$, and $\mathbf{x}^7$ need to move themselves forward to their prior NDLs.

Based on the above discussions, we notice that the addition and elimination of a solution usually does not shake the entire NDL structure of the current population. On the contrary, only
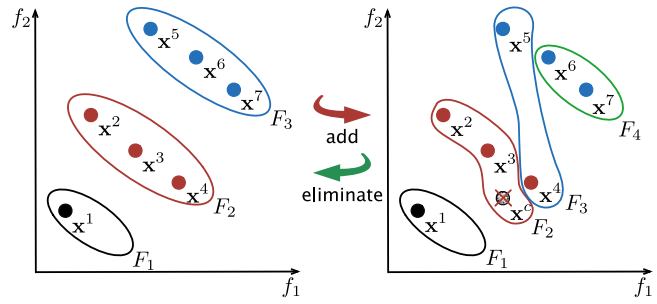
a limited number of solutions need to update their NDLs. Therefore, it is unnecessary to perform the NDS from scratch at each iteration of a steady state EMO algorithm. Instead, we only need to figure out the following three questions when a new candidate solution $\mathbf{x}^c$ comes in.

1) Which NDL does $\mathbf{x}^c$ belong to.
2) Is there any solution in P that needs to change its NDL.
3) If yes, what is the new NDL such solution belongs to.

Analogously, after eliminating an inferior solution by the environmental selection, we need to figure out the following two questions to update the NDL structure of the newly formed population.

1) Is there any solution in the newly formed P that needs to change its NDL.
2) If yes, what is the new NDL such solution belongs to.

In the next section, we will illustrate our ENLU method for addressing the above mentioned considerations.

III. EFFICIENT NONDOMINATION LEVEL UPDATE METHOD

Instead of performing the NDS from scratch, the ENLU method takes advantages of the existing knowledge of the current population to update the NDL structure. As discussed in Section II, the NDL structure might be changed both when we add a new candidate solution after reproduction and eliminate an inferior one after environmental selection. Bearing these two scenarios in mind, we will illustrate the technical details of the ENLU method step by step in the following paragraphs.

A. ENLU Method After Reproduction

According to the discussions in Section II, we have to figure out the following three issues.

1) Which NDL Does $\mathbf{x}^c$ Belong to: Here, we suggest a top-down approach to identify the NDL to which $\mathbf{x}^c$ belongs. More specifically, starting from F_1, we perform a pair-wise dominance comparison between $\mathbf{x}^c$ and all solutions in F_1. If $\mathbf{x}^c$ is nondominated with all solutions in F_1 or it dominates some ones therein, $\mathbf{x}^c$ is added to F_1. On the flip side, $\mathbf{x}^c$ does not belong to F_1 in case it is dominated by at least one solution in F_1. As long as such dominating solution is found, we do not compare the dominance relationship with the remaining solutions in F_1 any longer and turn to investigate the solutions

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

4   IEEE TRANSACTIONS ON CYBERNETICS

Algorithm 2: ENLU Method After Reproduction

Input:
- NDL structure $F = \{F_1, \cdots, F_l\}$
- offspring solution $\mathbf{x}^c$

Output: updated NDL structure F

1   $T \leftarrow \{\mathbf{x}^c\}$;
2   **for** $i \leftarrow 1$ **to** l **do**
3   **if** `CASE I` **then**
4   **continue**;
5   **else if** `CASE II` **then**
6       $F_i \leftarrow F_i \bigcup T$;
7   **break**;
8   **else if** `CASE III` **then**
9   Move all solutions in F_k, $k \in \{i, \cdots, l\}$, to F_{k+1};
10      $F_i \leftarrow T$;
11   **break**;
12   **else** // `CASE IV`
13      $F_i \leftarrow F_i \bigcup T$;
14      $T \leftarrow$ solutions in $F_i \wedge$ dominated by those in T;

15   **if** $i = l+1$ **then**
16     $F_{l+1} \leftarrow T$;

17   **return** F

Fig. 4.   Example of `CASE II` in ENLU method after reproduction.

Fig. 5.   Example of `CASE III` in ENLU method after reproduction.

in F_2, so on and so forth. Note that if $\mathbf{x}^c$ does not belong to any existing NDL F_i, where $i \in \{1, \ldots, l\}$, $\mathbf{x}^c$ is added to a newly created NDL F_{l+1}.

2) Is There Any Solution That Needs to Change Its NDL: According to the discussions in Section III-A1, for solutions in F_i, where $i \in \{1, \ldots, l\}$, only those dominated by the newly added solutions need to change their NDLs.

3) What Is the New NDL Such Solution Belongs to: Assume that $\mathbf{x}^l$ is going to be added to F_i, where $i \in \{1, \ldots, l\}$, and $\mathbf{x}^l$ dominates one or more solutions in F_i. These dominated solutions should be moved to another NDL after adding $\mathbf{x}^l$. According to the property of NDL, in case $j < i$ and $i, j \in \{1, \ldots, l\}$, none of these dominated solutions can dominate any solution in F_j, and each of them should be at least dominated by one solution in F_j. Therefore, these dominated solutions cannot be moved to an NDL prior to F_i. Moreover, each of these dominated solutions either be non-dominated or dominates a solution in F_{i+1}. In this case, it contradicts the property of NDL if those dominated solutions are moved to F_k, where $k > i + 1$. In summary, solutions in F_i and are dominated by $\mathbf{x}^l$ can only be moved from F_i to F_{i+1}.

Based on the above discussions, Algorithm 2 presents the pseudo-code of the ENLU method after reproduction, i.e., when $\mathbf{x}^c$ comes in. Note that the NDL structure of the parent population P is already known *a priori*. This is guaranteed in the steady-state EMO, e.g., steady-state NSGA-II, since the NDS is performed at the initialization procedure and the NDL structure is updated as long as $\mathbf{x}^c$ comes in. To start with, the algorithm first checks whether there exists a solution in F_1 that dominates $\mathbf{x}^c$. As long as we find such solution, we start comparing $\mathbf{x}^c$ with solutions in F_2, so on and so forth.
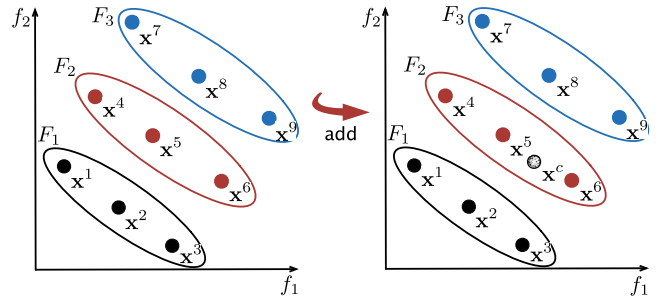
Generally speaking, we might meet one of the following four cases when checking with the solutions in F_i ($1 \leq i \leq l$).

1) `CASE I`: The newly added solutions[1] are dominated by at least one solution in F_i. According to the discussion in Section III-A3, `CASE I` only happens to $\mathbf{x}^c$. In particular, if $1 \leq i < l$, we stop comparing with the remaining solutions in F_i, and move to check with solutions in F_{i+1}. Otherwise, $\mathbf{x}^c$ is added to a newly created NDL F_{l+1}.

2) `CASE II`: The newly added solutions are nondominated with all solutions in F_i. In this case, the newly added solutions will be directly added to F_i, and no further comparison is required for the remaining NDLs. Fig. 4 presents a simple example to illustrate this case. Let us start the comparison from F_1. Since $\mathbf{x}^c$ is dominated by $\mathbf{x}^1$, it does not belong to F_1. Then, we move to check with solutions in F_2. Since $\mathbf{x}^c$ is nondominated with all solutions in F_2, it is added to F_2 and we stop comparing with the remaining solutions in F_3.

3) `CASE III`: The newly added solutions dominate all solutions in F_i. In this case, all solutions in F_k, where $k \in \{i, \ldots, l\}$, are moved to F_{k+1}, and the newly added solutions are added to F_i. Fig. 5 presents a simple example to illustrate this case. Let us start the comparison from F_1. Since $\mathbf{x}^c$ is dominated by $\mathbf{x}^1$, it does not belong to F_1. Then, we move to check with solutions in F_2. Since $\mathbf{x}^c$ dominates all solutions in F_2, it is added to F_2. In the meanwhile, solutions originally in F_2 and F_3 are, respectively, moved to F_3 and F_4.

---

[1]The newly added solution is $\mathbf{x}^c$ at the outset, and will be the solutions that need to change their NDLs thereafter.
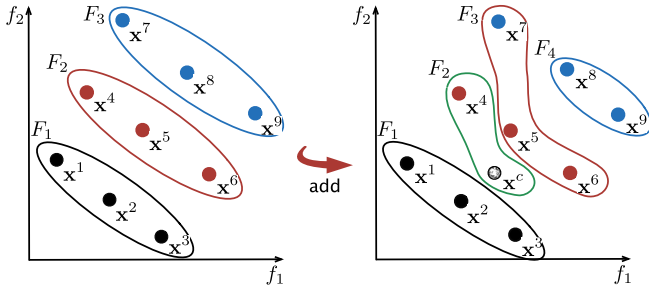
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LI *et al.*: ENLU METHOD FOR STEADY-STATE EMO

5

Fig. 6.   Example of CASE IV in ENLU method after reproduction.

Fig. 7.   Example of ENLU method after environmental selection.

4) CASE IV: The newly added solutions dominate one or more solutions in F_i. In this case, the newly added solutions, denoted as T in Algorithm 2, are added to F_i. In the meanwhile, the solutions originally in F_i and dominated by one or more solutions in T are used to form the new T for the next NDL. Fig. 6 presents a simple example to illustrate this case. Let us start the comparison from F_1. Since $\mathbf{x}^c$ is dominated by $\mathbf{x}^2$, it does not belong to F_1. Then $\mathbf{x}^c$ is compared with solutions in F_2. Since $\mathbf{x}^c$ dominates $\mathbf{x}^5$ and $\mathbf{x}^6$ and is nondominated with others, it is added to F_2 while $\mathbf{x}^5$ and $\mathbf{x}^6$ need to move to F_3. In F_3, since $\mathbf{x}^5$ and $\mathbf{x}^6$ dominate $\mathbf{x}^8$ and $\mathbf{x}^9$, $\mathbf{x}^5$ and $\mathbf{x}^6$ are added to F_3. At the same time, $\mathbf{x}^8$ and $\mathbf{x}^9$ are added to a newly created NDL F_4.

B. ENLU Method After Environmental Selection

According to the discussions in Section II, we have to figure out the following two issues.

1) Is There Any Solution That Needs to Change Its NDL: Let us assume that $\mathbf{x}^E$, which belongs to F_i, where $i \in \{1, \ldots, l\}$, is eliminated by the environmental selection. Note that solutions in F_j, where $1 \leq j \leq i$, either are nondominated with $\mathbf{x}^E$ or dominate it. Thus, the elimination of $\mathbf{x}^E$ cannot influence the NDL structure prior to F_i. Only solutions dominated by $\mathbf{x}^E$ might change their NDLs.

2) What Is the New NDL Such Solution Belongs to: Similar to the discussions in Section III-A, a solution can only move forward one NDL. Let us explain this by induction. Suppose that $\exists \mathbf{x}^* \in F_{i+1}$ and $\mathbf{x}^E \preceq \mathbf{x}^*$. $\exists \mathbf{x}' \in F_j$, where $1 \leq j \leq i-1$, and $\mathbf{x}' \preceq \mathbf{x}^E$. According to the transitivity property of the Pareto dominance, we have $\mathbf{x}' \preceq \mathbf{x}^*$. Therefore, $\mathbf{x}^*$ cannot be added to F_j. On the other hand, $\mathbf{x}^*$ can be added to F_i if and only if $\nexists \mathbf{x}'' \in F_i$ that $\mathbf{x}'' \preceq \mathbf{x}^*$.

Based on the above discussions, Algorithm 3 gives the pseudo-code of the ENLU method after environmental selection. To start with, we locate the NDL F_i to which $\mathbf{x}^E$ belongs (line 1 in Algorithm 3). Then, we identify the solutions in F_{i+1} and are dominated by $\mathbf{x}^E$. If there does not exist such solutions, the ENLU method terminates and no solution needs to change its NDL. Otherwise, we store the dominated solutions into a temporary archive S (line 3 in Algorithm 3). For each solution $\mathbf{x}$ in S, we compare the dominance relationship with the survived solutions in F_i. The solutions in S and dominated by the survived solutions in F_i are stored into a temporary archive D (line 5 in Algorithm 3),
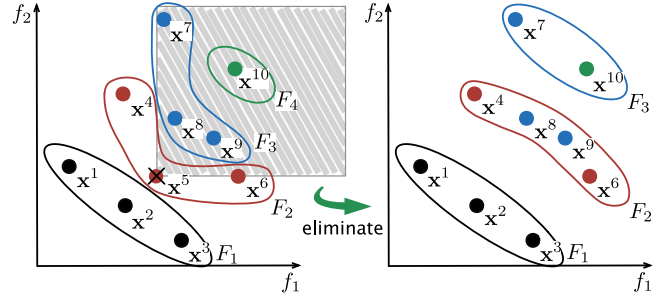

Algorithm 3: ENLU After Environmental Selection

Input:
 - NDL structure $F = \{F_1, F_2, \cdots, F_l\}$
 - eliminated solution $\mathbf{x}^E$

Output: updated NDL structure F

1 Locate the NDL F_i to which $\mathbf{x}^E$ belongs;
2 **while** $i < l$ **do**
3 $\quad$ $S \leftarrow$ solutions in $F_{i+1} \wedge$ dominated by $\mathbf{x}^E$;
4 $\quad$ **if** $S \neq \emptyset$ **then**
5 $\quad\quad$ $D \leftarrow$ solutions in $S \wedge$ dominated by the survived solutions in F_i;
6 $\quad\quad$ **if** $D = S$ **then**
7 $\quad\quad\quad$ **break**;
8 $\quad\quad$ $F_i \leftarrow F_i \bigcup S \backslash D$;
9 $\quad$ $i{+}{+}$;
10 **return** F

---

whereas those are nondominated with the survived solutions in F_i are added into this NDL (line 9 in Algorithm 3). If none of the solution in S can be added into F_i, we stop considering solutions after F_{i+1} (lines 6–8 in Algorithm 3). Note that if $\mathbf{x}^E \in F_l$, no more operation is required. Fig. 7 presents a simple example to illustrate the ENLU method after environmental selection. Suppose that $\mathbf{x}^5$ is eliminated from the population. Since all solutions in F_3 are dominated by $\mathbf{x}^5$, all of them have the chance to be added to F_2. We compare the dominance relationship between solutions in F_3 with $\mathbf{x}^4$ and $\mathbf{x}^6$, and we find that $\mathbf{x}^7$ is dominated by $\mathbf{x}^4$. Therefore, only $\mathbf{x}^8$ and $\mathbf{x}^9$ can be added to F_2. Afterwards, we find that $\mathbf{x}^{10} \in F_4$ is dominated by $\mathbf{x}^5$. Thus, we need to consider the movement of $\mathbf{x}^{10}$ from F_4 to F_3. Since $\mathbf{x}^{10}$ is nondominated with $\mathbf{x}^7$, it is added to F_3. At last, the ENLU method terminates.

IV. COMPUTATIONAL COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of the proposed ENLU method. As discussed in Section III, the ENLU method is performed twice at each iteration of a steady-state EMO algorithm (lines 3–5 in Algorithm 1). In the following paragraphs, we consider the computational complexity in two different scenarios, i.e., the ENLU method after reproduction and environmental selection, respectively.

A. Best-Case Complexity of ENLU Method

Let us first consider the scenario of the ENLU method after reproduction. The best-case happens when F_1 only contains a single solution and it is nondominated with the newly generated offspring solution $\mathbf{x}^c$. In this case, the ENLU method, shown in Algorithm 2, only requires one dominance comparison, i.e., m objective function comparisons. Thus, the best case complexity of the ENLU method after reproduction is $\mathcal{O}(m)$. As for the scenario of the ENLU method after environmental selection, the best-case happens when the elimination takes place at F_l. In this case, since this eliminated solution does not dominate any other in the population, the ENLU method, shown in Algorithm 3, does not require any further dominance comparison. Note that in the steady-state NSGA-II, this best-case always happens since its environmental selection deletes the worst solution from F_l as shown in line 5 of Algorithm 1. Nevertheless, there are some other steady-state EMO algorithm, e.g., our recently proposed one for many-objective optimization [16] in which the elimination of an inferior solution might not always happen in F_l. In summary, the best-case complexity of ENLU method is $\mathcal{O}(m)$.

B. Worst-Case Complexity of ENLU Method

The analysis of worst-case complexity is much more complicated. Let us still first consider the scenario of the ENLU method after reproduction.

Lemma 1: Given a population having N solutions, which form l $(1 \le l \le N)$ NDLs, i.e., $F_1, \ldots, F_l$. Each F_i contains φ_i $(1 \le \varphi_i \le N)$ solutions, where $i \in \{1, \ldots, l\}$, and $\sum_{i=1}^{l} \varphi_i = N$. The largest number of comparisons (NoCs) is calculated as

$$\text{NoC} = \varphi_1 + \sum_{i=2}^{k}(\varphi_{i-1} - 1)\varphi_i \tag{2}$$

where $k = l$ in case there does not exist any NDL, before F_l, in which the newly added solutions from the previous NDL dominate or are nondominated with all solutions; otherwise k is the index of the first such NDL.

The proof of Lemma 1 can be found in Appendix A in the supplementary material. It is the foundation to figure out the NDL structure that maximizes the NoC under the given N.

Lemma 2: When $l = 2$, the NDL structure $\varphi_1 = [(N/2)]+1$ and $\varphi_2 = N-[(N/2)]-1$ maximizes NoC, where $[*]$ can either be a rounded up or rounded down operation in case $(N/2)$ is not an integer.

The proof of Lemma 2 can be found in Appendix B in the supplementary material. Unfortunately, it is far from trivial to directly derive the NDL structure that maximizes the NoC when $l > 2$. In order to find some patterns, for a given N and l, we perform an exhaustive search to find the combinations of φ_i, where $i \in \{1, \ldots, l\}$, that give us the largest NoC. Due to the huge volume of different combinations, which grows exponentially with the increase of N and l, here, we set $N = 30$ as a constant and l varies from 3 to 6 in our experiment for illustrative purpose. Specifically, we have the following results.

1) When $l = 3$, there is one NDL structure that gives the largest NoC

φ_1	φ_2	φ_3
15	14	1

.

2) When $l = 4$, there are two different NDL structures that give the largest NoC

φ_1	φ_2	φ_3	φ_4
14	14	1	1
15	13	1	1

.

3) When $l = 5$, there is one NDL structures that gives the largest NoC

φ_1	φ_2	φ_3	φ_4	φ_5
14	13	1	1	1

.

4) When $l = 6$, there are two different NDL structures that give the largest NoC

φ_1	φ_2	φ_3	φ_4	φ_5	φ_6
13	13	1	1	1	1
14	12	1	1	1	1

.

Accordingly, we calculate the corresponding largest NoC achieved by different number of NDLs as follows:

$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$
226	224	209	195	181

.

Based on the above results, we have the following two observations.
1) For a given N and l, most solutions should be located in the first two NDLs in order to maximize NoC.
2) For a given N, the largest NoC decreases with the increase of l.

Lemma 3: When $l \ge 3$, the NDL structure $\varphi_1 = [((N - l + 3)/2)]$, $\varphi_2 = N - [((N - l + 3)/2)] - 1$, $\varphi_i = 1$, $i \in \{3, \ldots, l\}$ maximizes NoC, where $[*]$ can either be a rounded up or rounded down operation in case $((N - l + 3)/2)$ is not an integer.

The proof of Lemma 3 can be found in Appendix C in the supplementary material. This lemma provides the theoretical support to the first observation in the above exhaustive search, and it also gives the corresponding NDL structure that maximizes the NoC in a general case.

Theorem 1: For a given N, $l = 2$ maximizes NoC.

The proof of Theorem 1 can be found in Appendix D in the supplementary material. This theorem gives the theoretical support to the second observation in the above exhaustive search. Based on Lemma 2 and Theorem 1, we find that the worst-case complexity of our proposed ENLU method after reproduction is $\mathcal{O}(mN^2)$. Obviously, the computational complexity of the ENLU method after environmental selection cannot be larger than $\mathcal{O}(mN^2)$, even if an exhaustive search is performed. Therefore, we do not discuss the complexity therein. In summary, the worst-case complexity of ENLU method is $\mathcal{O}(mN^2)$.

V. EMPERIMENTAL RESULTS

The empirical studies in this paper consist of two parts. In the first part, we compare the performance of the ENLU method with five popular NDS algorithms on two different synthetic data sets; in the second part, we incorporate the ENLU method and the other five NDS algorithms into the EMO algorithm and compare their performance on some real optimization scenarios. In particular, we employ the number of objective comparisons as the indicator to evaluate the performance of different algorithms.[2] In order to mimic the reproduction, a point, randomly sampled from $[0, 1]^m$, is added

[2]Since different algorithms are implemented in different programming languages, we do not use CPU time cost in comparisons.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LI *et al.*: ENLU METHOD FOR STEADY-STATE EMO

7

to a data set before performing the ENLU method after reproduction; while for the environmental selection, a randomly chosen point is eliminated from the data set before performing the ENLU method after environmental selection. In addition, each NDS algorithm is launched 21 independent times for each data set. The median indicator values are used for comparisons. In the second part, we have implemented six steady-state NSGA-II variants by using the ENLU method and the other five NDS algorithms, respectively. The performance of different variants is studied on a variety of DTLZ problems with various number of objectives. In the following paragraphs, we at first give some brief descriptions on the five NDS algorithms and the implementations of two different synthetic data sets. Afterwards, we will discuss the experimental results in detail.

A. Nondominated Sorting Algorithms

1) Fast NDS [2]: Each solution is compared with other solutions in the population, and solutions that are nondominated with others are assigned to F_1. Then, solutions in F_1 are temporarily removed from the population, and the remaining nondominated solutions are assigned to F_2, so on and so forth. It is worth noting that, in order to reduce some unnecessary comparisons, the comparison between any two solutions only performs once.

2) Deductive Sort [5]: In order to reduce unnecessary comparisons, DS has two major strategies: one is to ignore the comparisons of dominated solutions to the others; the other is to infer the dominance relationship from the previous comparison records.

3) Corner Sort [6]: Its basic idea is to use the nondominated solutions to ignore their dominated solutions. It has two major strategies to reduce unnecessary comparisons: one is to ignore the dominated solutions as in DS; the other is to identify the nondominated solutions that are unique for CS.

4) Efficient Nondominated Sort [7]: In efficient nondominated sort (ENS), the comparison between any two solutions is at most once, thereby avoiding many unnecessary comparisons. It has two implementations: one uses a sequential search strategy (ENS-SS) and the other uses a binary search strategy (ENS-BS) to identify the NDL to which a solution belongs.

Our proposed ENLU method and FNDS are implemented in JAVA under the jMetal framework [21], an open source EMO algorithm package. The source codes of the other four NDS algorithms are obtained from their corresponding authors. Specifically, DS and CS are implemented in C++; ENS-BS and ENS-SS are implemented in MATLAB.

B. Synthetic Data Sets

1) Cloud Data Set: This data set contains solutions whose objective values are randomly sampled from a uniform distribution within the range $[0, 1]$. This randomly sampled population is unstructured, and it consists of solutions arranged in a random order. In addition, the randomly sampled population contains a varying number of NDLs, and each solution dominates an unpredictable number of solutions in the population. This data set tends to mimic the population structure in

the early stages of EMO, and it investigates the general ability to identify the NDL structure in a mixed population.

2) Fixed Fronts Data Set: This data set contains a population where solutions are divided into a controllable number of NDLs. Each NDL has almost the same size, and solutions in each NDL are distributed on a line or a hyper-plane. More detailed descriptions on the construction of this kind of data sets can be found in [5]. The fixed front data set tends to investigate the change of the computational cost with the variation of the number of NDLs. Note that the number of NDLs diminishes with the progress of evolution. Due to the page limit, further discussions upon this issue can be found in Appendix E in the supplementary material.

C. Experiments on Cloud Data Set

In this section, we test the performance of the ENLU method with the other five NDS algorithms on cloud data sets in two-, five-, ten-, and fifteen-objective cases, respectively. For each case, the size of a data set ranges from 100 to 5000 with an increment of 100. That is to say, for a given number of objectives, there are 50 randomly generated populations in total for the empirical studies.

Fig. 8 plots the variations of the number of objective comparisons for different data set sizes. Note that the *y*-axes of Fig. 8 are labeled in log-scale, since FNDS costs much more objective comparisons than others. It is worth noting that the number of objective comparisons of FNDS increases with the growth of the data set size, whereas its trajectories have little change for different number of objectives. This can be explained as the computational cost of FNDS largely depends on the population size. Since DS ignores some dominated solutions in sorting, it requires fewer comparisons than FNDS. As discussed in [7], in ENS-SS and ENS-BS, only solutions, which have already been assigned an NDL, are used to compare with the other unassigned ones. Empirical results in Fig. 8 demonstrate that both ENS-SS and ENS-BS indeed reduce many unnecessary comparisons. Especially for the two-objective case, ENS-BS requires much fewer objective comparisons than the other four NDS algorithms. However, in five- and ten-objective cases, ENS-SS performs slightly better than ENS-BS. In addition, we notice that the number of objective comparisons of DS, ENS-SS, and ENS-BS increases with the growth of dimensionality. Even worse, as shown in Fig. 8, the performance of these three algorithms almost degenerate to FNDS in the ten- and fifteen-objective cases. As for CS, it takes the advantage of the corner solution, which has the best value in a particular objective function, to reduce unnecessary comparisons. In contrast to the $m(N - 1)$ objective comparisons for identifying a nondominated solution, the identification of a corner solution only requires $N - 1$ objective comparisons. This property makes CS very efficient for the many-objective scenario. From the results shown in Fig. 8, we find that the performance of CS is only better than FNDS in the two-objective case, whereas it performs better than the other four NDS algorithms when the number of objectives becomes large. Nevertheless, the ENLU method shows a constantly best performance in all comparisons. Its superiority

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

8   IEEE TRANSACTIONS ON CYBERNETICS
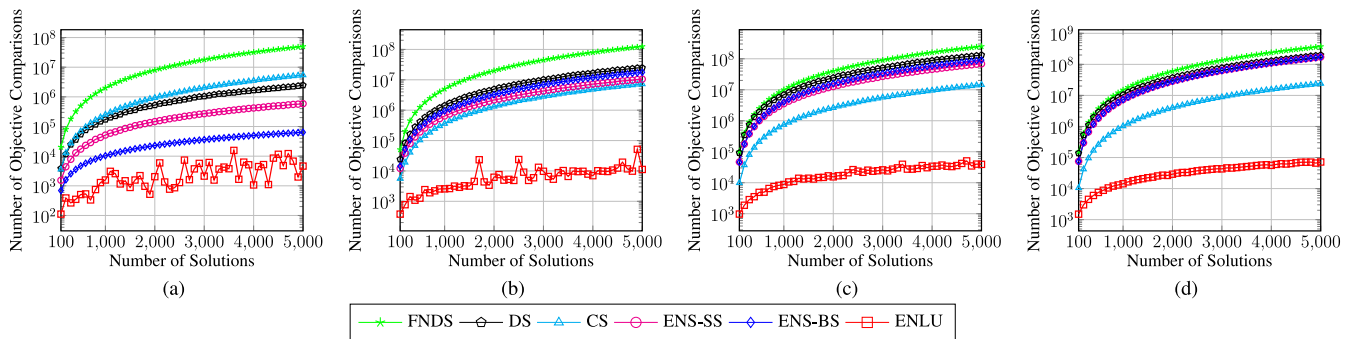

Fig. 8. Median number of objective comparisons of ENLU method and the other five NDS algorithms for cloud data sets. (a) Two-objective case. (b) Five-objective case. (c) Ten-objective case. (d) Fifteen-objective case.
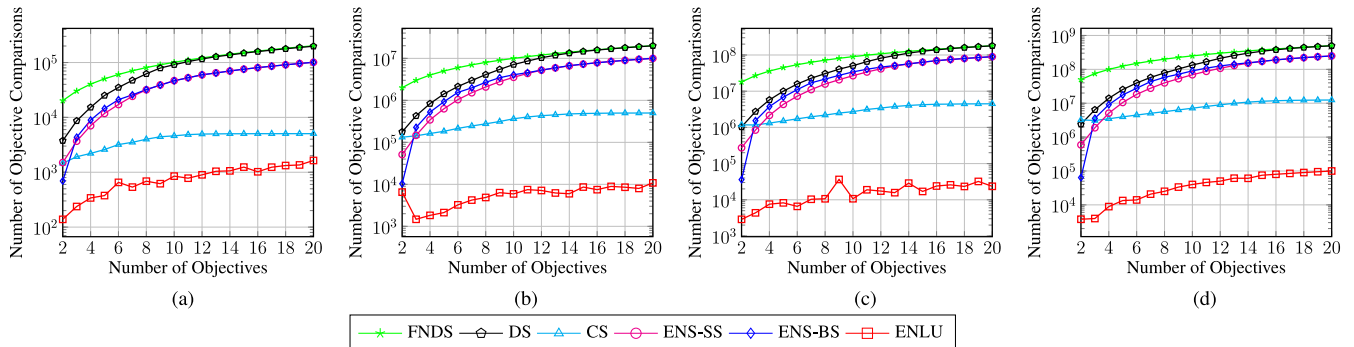

Fig. 9. Median number of objective comparisons of ENLU method and the other five NDS algorithms for cloud data sets with fixed sizes. (a) $N = 100$. (b) $N = 1000$. (c) $N = 3000$. (d) $N = 5000$.

becomes even more significant with the increase of the number of objectives. It is interesting to note that the trajectories of the ENLU method fluctuate significantly in the two- and five-objective cases, and become stable later on. As discussed in Section IV, the computational cost of the ENLU method largely depends on the population distribution. In the low-dimensional case, the NDL structure is rather chaotic, thereby adding a new solution might largely shake the original NDL structure. On the other hand, the number of NDLs diminishes with the growth of dimensionality, which makes the NDL structure become relatively simpler. Thereby, the number of objective comparisons cost by the ENLU method becomes stable in the high-dimensional cases. Due to the page limit, the issues of NDL structure will be further explored in Appendix E in the supplementary material.

In the above experiments, we investigate the performance variation for different data set sizes for a particular dimensionality. People may also interest in the performance variation on a data set with a fixed size in various dimensionalities. To this end, we conduct another set of experiments on some cloud data sets with a fixed size (100, 1000, 3000, and 5000, respectively), where the number of objectives varies from 2 to 20 for each case. Fig. 9 presents the performance comparisons of ENLU method and the other five NDS algorithms. From these experimental results, we have observed a similar trend as in Fig. 8: the performance of DS, ENS-SS, and ENS-BS gradually degenerate to FNDS with the growth of dimensionality. In particular, the number of objective comparisons cost by DS becomes the same as FNDS in case more than fifteen

objectives have been considered. This can be explained as the cloud data sets with more than fifteen objectives usually have only one NDL, thus no dominated solutions can be ignored by the DS. As for CS, the number of objective comparisons slightly increases with the growth of dimensionality. And similar to the observations in Fig. 8, CS costs more objective comparisons than DS, ENS-SS and ENS-BS when the number of objectives is small. However, with the increase of the number of objectives, CS shows constantly better performance than the other NDS algorithms. Nevertheless, as expected, our proposed ENLU method is the most efficient method, which costs much less number of objective comparisons, comparing to all other NDS algorithms.

D. Experiments on Fixed Fronts Data Sets

After the experiments on cloud data sets, this section investigates the performance of ENLU method and the other five NDS algorithms on data sets with a controllable number of NDLs. In particular, we consider two factors that might influence of the computational cost, i.e., the number of NDLs and the number of objectives.

The first experiment investigates the performance of ENLU method and the other five NDS algorithms on the fixed fronts data sets with two, five, ten, and fifteen objectives. The population size is fixed to 2000, and the number of NDLs varies from 2 to 70 with an increment of one, resulting in 69 populations in total for each test case. Fig. 10 presents the comparison results of the ENLU method with FNDS, DS, CS, ENS-SS,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.
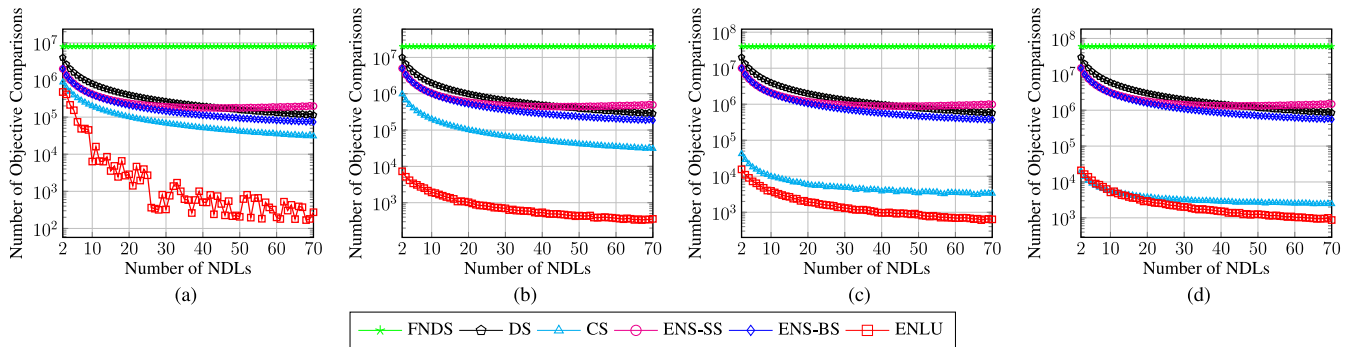
LI *et al.*: ENLU METHOD FOR STEADY-STATE EMO

9

Fig. 10. Median number of objective comparisons of ENLU method and the other five NDS algorithms for fixed fronts data sets with controllable number of NDLs. (a) Two-objective case. (b) Five-objective case. (c) Ten-objective case. (d) Fifteen-objective case.
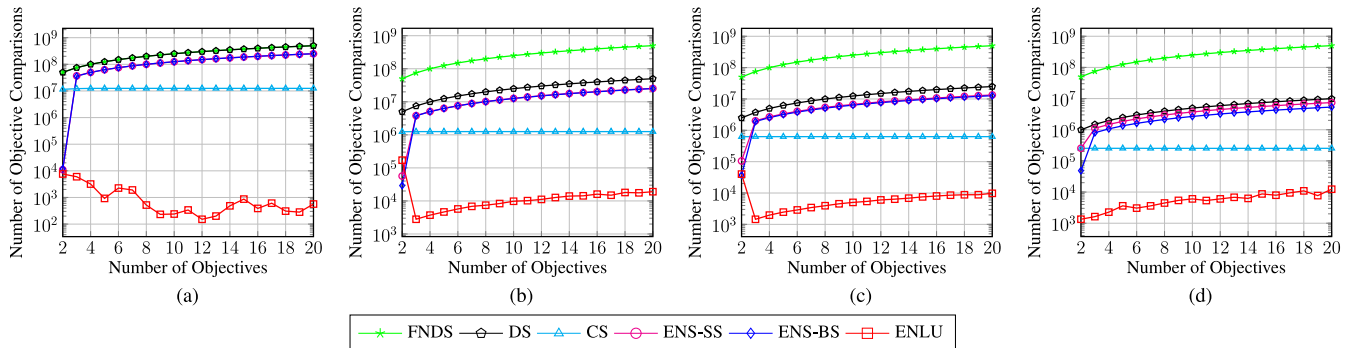

Fig. 11. Median number of objective comparisons of ENLU method and the other five NDS algorithms for fixed fronts data sets with fixed number of NDLs. (a) $l = 1$. (b) $l = 10$. (c) $l = 30$. (d) $l = 50$.

and ENS-BS, regarding the number of objective comparisons. Similar to the observations for the cloud data sets, as shown in Fig. 10, FNDS costs the largest number of objective comparisons among all six algorithms. In addition, it is also interesting to note that the trajectories of FNDS keep stable over different number of NDLs. This can be explained as the number of objective comparisons incurred by FNDS largely depends on the number of objectives and population size. As discussed in [5], this quantity is $m \times (N^2 - N)$, regardless of the number of NDLs. In contrast to FNDS, a significant decrease in the number of objective comparisons has been witnessed by the other five algorithms. Generally speaking, their trajectories share a common trend where the number of objective comparisons decreases with the increase of the number of NDLs. More specifically, the performance of ENS-SS is similar to ENS-BS when the number of NDLs is relatively small, whereas its performance deteriorates with the increase of the number of NDLs. Even worse, ENS-SS costs more objective comparisons than DS when the number of NDLs is larger than 40. As for CS, it costs less number of objective comparisons when the number of objectives becomes large. All in all, our proposed ENLU method is the best algorithm in most test cases.

In the second experiment, we test the performance of the ENLU method and the other five NDS algorithms on data sets with 1, 10, 20, and 50 NDLs, respectively, for different number of objectives. Here, the population size is constantly set as 5000, and the number of objectives varies from 2 to 20 with an increment of one. From the empirical results shown in Fig. 11, we find that our proposed ENLU method is the best

candidate in most test cases. Although ENS-SS and ENS-BS cost fewer objective comparisons in two-objective case, their trajectories surge up toward a high level later on. It is worth noting that the performance of DS is almost the same as FNDS when there is only one NDL. This is because DS cannot ignore any solution when all solutions are nondominated with each other. As for CS, its required number of objective comparisons keep stable all the time.

E. Performance Investigations in Steady-State NSGA-II

Other than the empirical studies on synthetic data sets, it is also interesting to see the efficiency improvement when the ENLU method is embedded in a steady-state EMO algorithm. To this end, we develop six steady-state NSGA-II variants. In particular, the pseudo-code of the variant that uses the ENLU method to update the NDL structure is given in Algorithm 4, while the other variants are, respectively, using FNDS, DS, ENS-SS, ENS-BS, and CS methods to replace line 4 of Algorithm 1. DTLZ1 to DTLZ4 [20], with three, five, eight, ten, and fifteen objectives, are chosen as the benchmark problems. All steady-state NSGA-II variants use the simulated binary crossover [22] and polynomial mutation [23] for off-spring generation. The crossover probability is $p_c = 1.0$ and its distribution index is $\eta_c = 30$. The mutation probability is $p_m = 1/n$ and its distribution index is $\eta_m = 20$. According to our recent studies on many-objective optimization [16], the settings of the number of generations and population size for different number of objectives are given in Table I.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

10   IEEE TRANSACTIONS ON CYBERNETICS
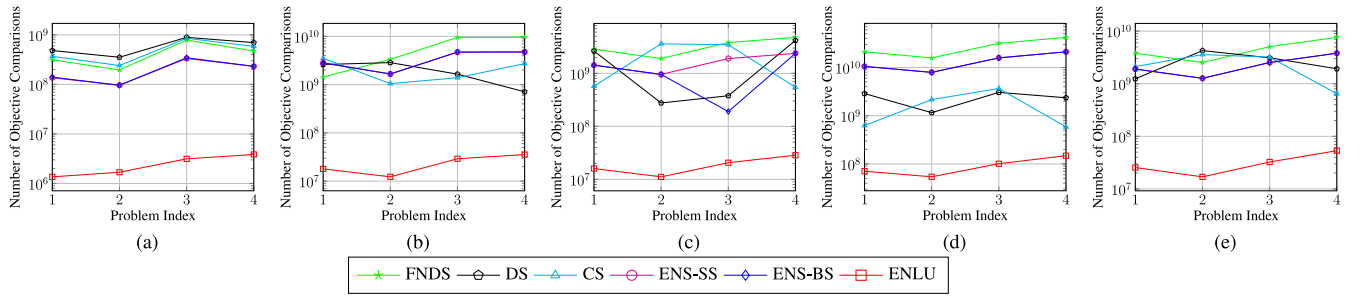

Fig. 12. Median number of objective comparisons of six steady-state NSGA-II variants by using ENLU method and the other five NDS algorithms on DTLZ1–DTLZ4. Note that problem index i means DTLZi. (a) Three-objective. (b) Five-objective. (c) Eight-objective. (d) Ten-objective. (e) Fifteen-objective.

Algorithm 4: Steady-State NSGA-II Using ENLU Method

Input: algorithm parameters
Output: population P
1 Initialize a population $P \leftarrow \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$;
2 Use NDS to divide P into several NDLs, i.e., $F_1, \cdots, F_l$;
3 **while** *termination criterion is not met* **do**
4   Mating selection and generate an offspring $\mathbf{x}^c$;
5   Use ENLU method to update the NDL structure of $P' \leftarrow P \bigcup \{\mathbf{x}^c\}$;
6   Identify the worst solution $\mathbf{x}'$ and set $P \leftarrow P' \setminus \{\mathbf{x}'\}$;
7   Use ENLU method to update the NDL structure of P;
8 **return** P;

TABLE I
NUMBER OF GENERATIONS FOR DIFFERENT TEST INSTANCES

Test Instance	$m = 3$	$m = 5$	$m = 8$	$m = 10$	$m = 15$
DTLZ1	400	600	750	1,000	1,500
DTLZ2	250	350	500	750	1,000
DTLZ3	1,000	1,000	1,000	1,500	2,000
DTLZ4	600	1,000	1,250	2,000	3,000
Population Size	92	212	156	276	136

Algorithm 5: Canonical NSGA-II Using ENLU Method

Input: algorithm parameters
Output: population P
1 Initialize a population $P \leftarrow \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$;
2 Use NDS to divide P into several NDLs, i.e., $F_1, \cdots, F_l$;
3 **while** *termination criterion is not met* **do**
4   Mating selection and generate the offspring population $Q \leftarrow \{\mathbf{x}_1^c, \cdots, \mathbf{x}_N^c\}$;
5     $P' \leftarrow P$;
6   **for** $i \leftarrow 1$ **to** N **do**
7         $P' \leftarrow P' \bigcup \{\mathbf{x}_i^c\}$;
8   Use ENLU method to update the NDL structure of P';
9     $i \leftarrow 0, P \leftarrow \emptyset$;
10   **while** $|P| < N$ **do**
11         $i \leftarrow i + 1, P \leftarrow P \bigcup F_i$;
12   **while** $|P| > N$ **do**
13   Identify the worst solution $\mathbf{x}' \in F_i$ and set $P \leftarrow P \setminus \{\mathbf{x}'\}$;
14   Use ENLU method to update the NDL structure of P;
15 **return** P;

Each steady-state NSGA-II variant is launched 21 independent times, and Fig. 12 presents the median number of objective comparisons cost by these six variants on different test instances. From the experimental results, we can clearly see that the steady-state NSGA-II with the ENLU method costs much fewer (more than 10 times) objective comparisons than the other five variants. Although FNDS always costs more objective comparisons than the other NDS methods in synthetic data sets, it is not the worst candidate when embedding in a steady-state NSGA-II in some cases. For instance, in the three-objective case, the steady-state NSGA-II variants with DS and CS consume more objective comparisons than the one using FNDS. It is interesting to note that the number of objective comparisons cost by ENS-SS and ENS-BS is almost the same in most cases. Furthermore, these two methods have shown better performance than the other NDS methods in the three-objective case. But their superiorities gradually vanish with the growth of dimensionality. In summary, our proposed ENLU method not only shows the best performance in synthetic data sets, it is also a reliable and efficient method to maintain NDL structure in a steady-state EMO algorithm.

F. Incorporation of ENLU Into the Generational Scenario

Although the ENLU method is designed for the steady-state evolution model, an interesting question is whether it is also useful for the generational scenario? To address this issue, we incorporate the ENLU method into the canonical NSGA-II whose pseudo-code is given in Algorithm 5. Note that each offspring is added to the parent population one by one, followed by the ENLU method for updating the NDL structure of the newly hybrid population each time (lines 6–9 in Algorithm 5). Accordingly, the truncation is also conducted in a sequential manner, coupled with the ENLU method to keep the NDL structure up to date (lines 14–17 in Algorithm 5). Similar to Section V-E, we develop five other canonical NSGA-II variants, which use FNDS, DS, ENS-SS, ENS-BS, and CS methods to perform the NDS, respectively. The experimental settings are exactly same as Section V-E, and Fig. 13 presents the median number of objective comparisons cost by these six variants on different test instances. From the
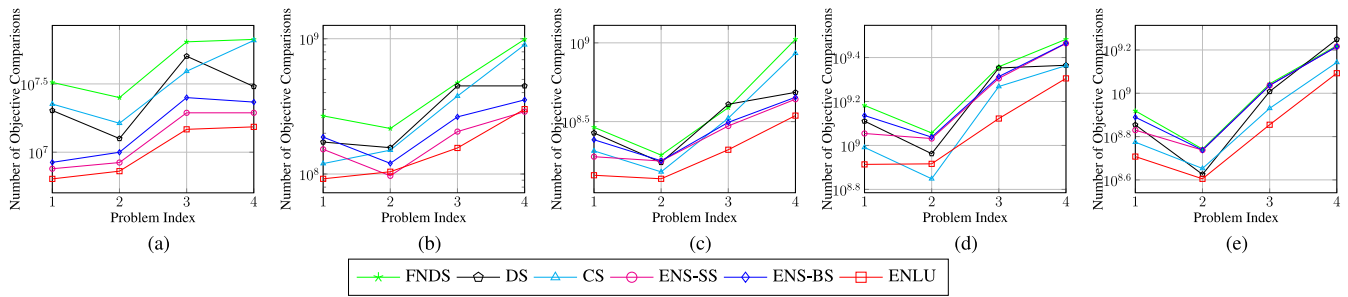
Fig. 13.   Median number of objective comparisons of six NSGA-II variants by using ENLU method and the other five NDS algorithms on DTLZ1–DTLZ4. Note that problem index *i* means DTLZ*i*. (a) Three-objective. (b) Five-objective. (c) Eight-objective. (d) Ten-objective. (e) Fifteen-objective.

experimental results, we can see that the canonical NSGA-II with the ENLU method costs the fewest number of objective comparisons than the other five variants in most cases. ENS-SS and ENS-BS are the second best NDS methods on three- and five-objective scenarios; while their performance deteriorate with the number of objectives. In contrast, the performance of CS and DS are not very promising in the low-dimensional scenarios; while their superiorities become evident when the number of objectives becomes large. Moreover, we notice that the superiority of our proposed ENLU method is not as much as that in the steady-state scenario. All in all, it is very interesting to see that the ENLU method is also useful for the generational evolution model. This suggests that maintaining the NDL structure without resorting to the NDS is general to both steady-state and generational evolution models.

VI. CONCLUSION

NDS, which is a basic step in EMO, can be very time consuming when the number of objectives and population size become large. To avoid unnecessary comparisons, instead of performing the NDS from scratch, this paper presents an ENLU method, which takes advantages of the current population to update the NDL structure, for the steady-state EMO. At each iteration, the ENLU method is performed twice: one is after reproduction and the other is after environmental selection. By leveraging the population structure, the ENLU method only updates the NDLs of a limited number of solutions. Theoretically, the best-case complexity of the ENLU method is $\mathcal{O}(m)$, while the worst-case complexity is $\mathcal{O}(mN^2)$. Although the proposed ENLU method is very simple and straightforward, extensive experiments have shown that it avoids a significant amount of unnecessary comparisons, not only in the synthetic data sets, but also in some real optimization scenarios. Furthermore, it is also very interesting to see that the ENLU method can also be useful for the generational evolution model. In future, we believe that heuristics (e.g., taking advantage of previous comparisons as done in [5]) and advanced data structures (e.g., K-d tree [24]) are worth being applied to the ENLU method to further improve its computational efficiency. It is also interesting to apply the ENLU method to both steady-state and generational EMO algorithms in more real optimization scenarios [25]–[27].

REFERENCES

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

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

[3] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, Oct. 1975.

[4] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 5, pp. 503–515, Oct. 2003.

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

[6] H. Wang and X. Yao, "Corner sort for Pareto-based many-objective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 92–102, Jan. 2014.

[7] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to non-dominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

[8] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba, "On the effect of the steady-state selection scheme in multi-objective genetic algorithms," in *Proc. 5th Int. Conf. Evol. Multi Criterion Optim. (EMO)*, Nantes, France, 2009, pp. 183–197.

[9] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[10] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.

[11] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.

[12] K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions," *Evol. Comput.*, vol. 13, no. 4, pp. 501–525, 2005.

[13] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 114–130, Feb. 2014.

[14] K. Nag, T. Pal, and N. R. Pal, "ASMiGA: An archive-based steady-state micro genetic algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 40–52, Jan. 2015.

[15] K. Li, S. Kwong, and K. Deb, "A dual-population paradigm for evolutionary multiobjective optimization," *Inf. Sci.*, vol. 309, pp. 50–72, Jul. 2015.

[16] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.

[17] A. J. Nebro and J. J. Durillo, "On the effect of applying a steady-state selection scheme in the multi-objective genetic algorithm NSGA-II," in *Nature-Inspired Algorithms for Optimisation* (Studies in Computational Intelligence), vol. 193. Heidelberg, Germany: Springer, 2009, pp. 435–456.

[18] M. Buzdalov and V. Parfenov, "Various degrees of steadiness in NSGA-II and their influence on the quality of results," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Madrid, Spain, 2015, pp. 749–750.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[19] M. Buzdalov, I. Yakupov, and A. Stankevich, "Fast implementation of the steady-state NSGA-II algorithm for two dimensions based on incremental non-dominated sorting," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Madrid, Spain, 2015, pp. 647–654.

[20] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization* (Advanced Information and Knowledge Processing), A. Abraham, L. Jain, and R. Goldberg, Eds. London, U.K.: Springer, 2005, pp. 105–145.

[21] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.

[22] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 1–34, 1994.

[23] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Informat.*, vol. 26, no. 4, pp. 30–45, 1996.

[24] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[25] K. Li *et al.*, "Achieving balance between proximity and diversity in multi-objective evolutionary algorithm," *Inf. Sci.*, vol. 182, no. 1, pp. 220–242, 2012.

[26] K. Li, S. Kwong, R. Wang, K.-S. Tang, and K.-F. Man, "Learning paradigm based on jumping genes: A general framework for enhancing exploration in evolutionary multiobjective optimization," *Inf. Sci.*, vol. 226, pp. 1–22, Mar. 2013.

[27] K. Nag and N. R. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 499–510, Feb. 2016.

Ke Li received the B.Sc. and M.Sc. degrees in computer science and technology from Xiangtan University, Xiangtan, China, in 2007 and 2010, respectively, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2014.

He was a Post-Doctoral Research Associate with Michigan State University, East Lansing, MI, USA, and a Research Fellow with the University of Birmingham, Birmingham, U.K. He is currently a Lecturer (Assistant Professor) with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, U.K. His current research interests include the evolutionary multiobjective optimization, large scale optimization, statistical machine learning, and applications in software engineering and industrial design.

Dr. Li has served as the regular Reviewer and has published several research paper in renowned journals such as the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.

Kalyanmoy Deb (F'13) received the bachelor's degree in mechanical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, and the master's and Ph.D. degrees from the University of Alabama, Tuscaloosa, AL, USA, in 1985, 1989, and 1991, respectively.

He is the Koenig Endowed Chair Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA, where he also holds joint appointments with the Department of Mechanical Engineering and with the Department of Computer Science and Engineering. He has published over 435 research papers with over 90 000 Google Scholar citation with an H-index of 100. His current research interests include evolutionary optimization and their applications in optimization, modeling, and machine learning.

Prof. Deb was a recipient of the Infosys Prize, the TWAS Prize in Engineering Sciences, the CajAstur Mamdani Prize, the Distinguished Alumni Award from IIT Kharagpur, the Edgeworth-Pareto Award, the Bhatnagar Prize in Engineering Sciences, and the Bessel Research Award from Germany. He is in the Editorial Board on 20 major international journals. He is a fellow of ASME, and three Indian science and engineering academies.

Qingfu Zhang (M'01–SM'06) received the B.Sc. degree in mathematics from Shanxi University, Taiyuan, China, in 1984, the M.Sc. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

He is a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, and a Changjiang Visiting Chair Professor with Xidian University. He holds two patents and has authored many research publications. His current research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications. He is currently leading the Metaheuristic Optimization Research Group with the City University of Hong Kong.

Dr. Zhang was a recipient of the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation 2009 for MOEA/D, multiobjective optimization algorithm developed in his group, the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, and the Highly Cited Researcher Award in Computer Science by the Web of Science in 2016. He is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS. He is also an Editorial Board Member of three other international journals.

Qiang Zhang received the B.Sc. and Ph.D. degrees in computer science from the City University of Hong Kong, Hong Kong, in 2009 and 2013, respectively.

He is a Post-Doctoral Research Associate with the Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome, Rome, Italy. His current research interests include algorithms, mechanism design, and optimization.