# Efficient Non-domination Level Update Method for Steady-State Evolutionary Multi-objective Optimization

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

**COIN Report Number 2015022**

**Abstract**

Non-dominated sorting, which divides a population into several non-domination levels, is a basic step in many evolutionary multi-objective optimization 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, non-dominated sorting 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 non-domination level update method to reduce the cost for maintaining the non-domination level structure in steady-state evolutionary multi-objective optimization. Rather than perform the non-dominated sorting from scratch, our method only updates the non-domination levels of a limited number of solutions by extracting the knowledge from the current non-domination level structure. Notice that our non-domination level 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 non-dominated sorting methods, our proposed method avoids a significant amount of unnecessary comparisons, not only in synthetic data sets, but also in real optimization scenarios.

**Index Terms**

Pareto dominance, non-domination level, non-dominated sorting, computational complexity, steady-state evolutionary multi-objective optimization

## I. Introduction

A multi-objective optimization problem (MOP) can be stated as follows:

$$\begin{aligned} &\text{minimize} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))^T \\ &\text{subject to} \quad \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

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 objective.

Non-dominated sorting (NDS) is a procedure that divides a population of solutions into several non-domination levels (NDLs) according to their dominance relationships. It gives a relative quality of solutions, belonging to a specific NDL, with respect to the others. Although NDS is a basic step in 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 NDS was recognized and addressed by Deb *et al.* in [2]. They developed the fast NDS method which avoids some unnecessary dominance comparisons by taking advantages of 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 a NDS method with a computational complexity of $\mathcal{O}(Nlog^{m-1}N)$, 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, to reduce the computational cost of NDS. Although these two methods hold the same worst-case complexity of $\mathcal{O}(mN^2)$ as the fast NDS, empirical studies showed that both of them outperform the fast NDS 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 number of objective comparisons in many-objective scenarios, Wang and Yao proposed a corner sort method [6]. Its basic

K. Li is with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK (e-mail: keli.genius@gmail.com)

K. Deb are 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, Kowloon, Hong Kong SAR (e-mail: qingfu.zhang@cityu.edu.hk), and the School of Electronic Engineering and Computer Science, University of Essex, UK (qzhang@essex.ac.uk)

Q. Zhang is with the Institute of Informatics, University of Warsaw, 02-097 Warsaw, Poland (e-mail: csqzhang@gmail.com)

**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 **end**
7 **return** $P$;

idea is to use the non-dominated 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 a NDL.

To our best knowledge, most, if not all, studies on NDS are discussed in the context of a generational evolution model, whereas few have considered the situation for a steady-state evolution model yet. In [8], Buzdalov *et al*. presented a incremental NDS for steady-state EMO. However, this method can only work for the two-dimensional case. The major difference between the generational evolution model and the steady-state evolution model is the moment to perform environmental selection. In the prior case, a population of offspring solutions are generated before competing with their parents; while in latter case, the parent population is updated once a new candidate solution has been generated. It is interesting to note that the steady-state EMO algorithm shows better performance, in terms of convergence and diversity, than its generational counterpart in some recent studies [9]–[11].

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 non-domination level update (ENLU) method to reduce the cost for maintaining the NDL structure in steady-state EMO. By using 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, 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, 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 fast NDS method, extensive experiments demonstrate that it avoids a significant amount of unnecessary comparisons in practice.

In the rest of this paper, we first discuss the motivations of this work 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) [2]. 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 NDS to divide the hybrid population $P'$, a combination of $P$ and $\mathbf{x}^c$, into $l$ ($l \geq 1$) NDLs, i.e., $F_1, \cdots, F_l$ (line 4 in Algorithm 1). More specifically, all non-dominated 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 $P' - F_1$ are assigned to $F_2$, so on and so forth. Note that each solution in $F_i$ is either non-dominated with or dominated by at least one solution in $F_j$, where $i > j$ and $i, j \in \{1, \cdots, l\}$. After the NDS procedure, 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 NDS requires pair-wise dominance comparisons among solutions, it often becomes the most time-consuming part compared to the rest of an EMO algorithm. To illustrate this problem, we perform two simple experiments by using steady-state NSGA-II on several DTLZ2 test instances [12]. 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 2,000 with a step size 100. The number of generations is set as

1,000 for all cases. From the empirical results shown in Fig. 1, we clearly see that NDS indeed consumes a dominating CPU time ratio (around 70%) compared to the other parts of the steady-state NSGA-II. 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 NDS (or in other words, maintaining the NDL structure), especially for a large number of objectives and population size.

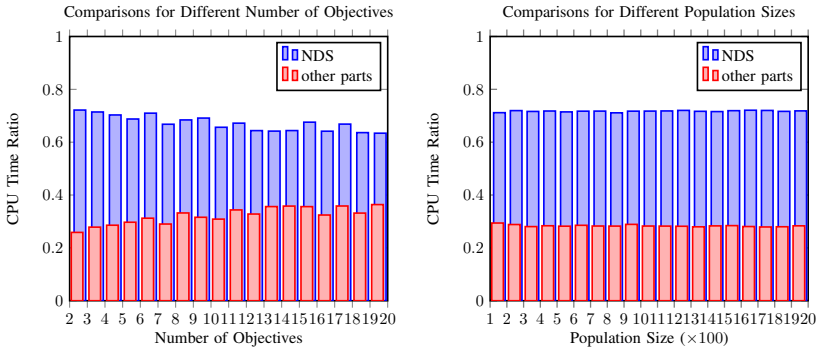


Fig. 1: The comparisons of CPU time ratios cost by NDS and the other parts of the steady-state NSGA-II.

To this end, an idea naturally comes out: is it really necessary to perform 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., $F_1 = \{\mathbf{x}^1\}, F_2 = \{\mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4\}, F_3 = \{\mathbf{x}^5, \mathbf{x}^6, \mathbf{x}^7\}$. 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 Fig. 2 and Fig. 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.

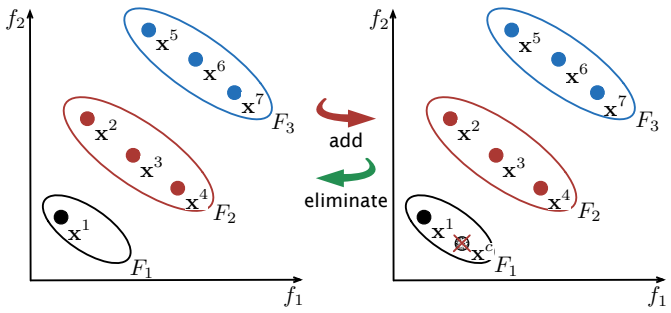


Fig. 2: The NDL structure keeps unchanged when $\mathbf{x}^c$ is added and eliminated.

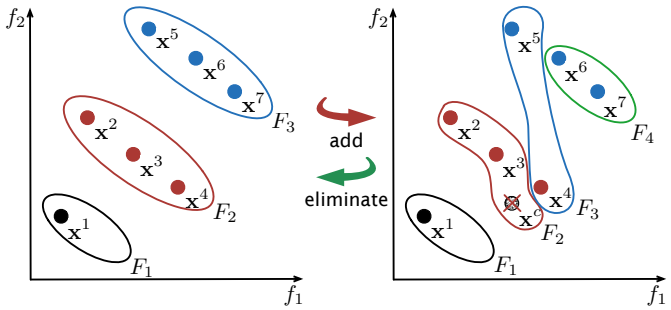


Fig. 3: $\mathbf{x}^4$, $\mathbf{x}^6$ and $\mathbf{x}^7$ need to change their NDLs when $\mathbf{x}^c$ is added and eliminated.

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 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 $\mathbf{x}^c$ belongs 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 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 in detail by addressing the above mentioned considerations.

## III. Efficient Non-domination Level Update Method

Rather than perform NDS from scratch, 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 our ENLU method step by step in the following paragraphs.

### A. ENLU Method After Reproduction

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

*1) Which NDL $\mathbf{x}^c$ belongs 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 non-dominated 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 in $F_2$, so on and so forth. Note that if $\mathbf{x}^c$ does not belong to any existing NDL $F_i$, where $i \in \{1, \cdots, l\}$, $\mathbf{x}^c$ 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 $i \in \{1, \cdots, l\}$, 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 $\mathbf{x}^{\mathrm{l}}$ is going to be added to $F_i$, where $i \in \{1, \cdots, l\}$, and $\mathbf{x}^{\mathrm{l}}$ dominates one or more solutions in $F_i$. These dominated solutions should be moved to another NDL after adding $\mathbf{x}^{\mathrm{l}}$. According to the property of NDL, in case $j < i$ and $i, j \in \{1, \cdots, l\}$, 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 a 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 $\mathbf{x}^{\mathrm{l}}$ can only be moved from $F_i$ to $F_{i+1}$.

Based on the above discussions, Algorithm 2 presents the pseudo-code of 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 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 non-dominated 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 non-dominated 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 $k \in \{i, \cdots, l\}$, 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.

**Algorithm 2:** ENLU method after reproduction

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

**Output**: updated NDL structure $F$

```
T ← {x^c};
for i ← 1 to l do
    if CASE I then
        continue;
    else if CASE II then
        F_i ← F_i ⋃ T;
        break;
    else if CASE III then
        Move all solutions in F_k, k ∈ {i, ··· , l}, to F_{k+1};
        F_i ← T;
        break;
    else // CASE IV
        F_i ← F_i ⋃ T;
        T ← solutions in F_i ∧ dominated by those in T;
    end
end
if i = l + 1 then
    F_{l+1} ← T;
end
return F
```

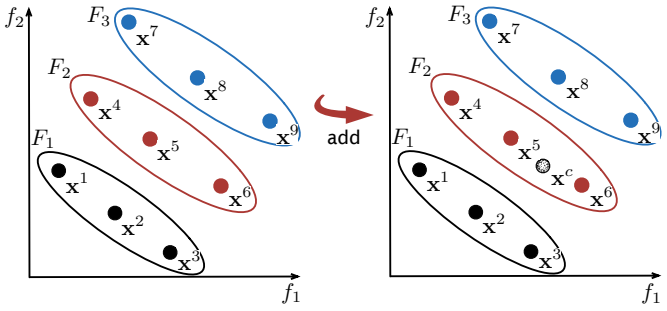


Fig. 4: An example of CASE II in ENLU method after reproduction.

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 are 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 non-dominated 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 discussion 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 $\mathbf{x}^{\mathsf{E}}$, which belongs to $F_i$, where $i \in \{1, \cdots, l\}$, is eliminated by the environmental selection. Note that solutions in $F_j$, where $1 \leq j \leq i$, either are non-dominated with $\mathbf{x}^{\mathsf{E}}$ or dominate it. Thus, the elimination of $\mathbf{x}^{\mathsf{E}}$ cannot influence the NDL structure prior to $F_i$. Only solutions dominated by $\mathbf{x}^{\mathsf{E}}$ 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 $\mathbf{x}^{\mathsf{E}} \preceq \mathbf{x}^*$. $\exists \mathbf{x}' \in F_j$, where $1 \leq j \leq i-1$,

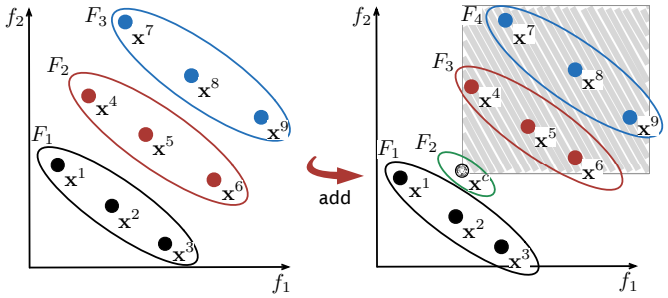


Fig. 5: An example of CASE III in ENLU method after reproduction.

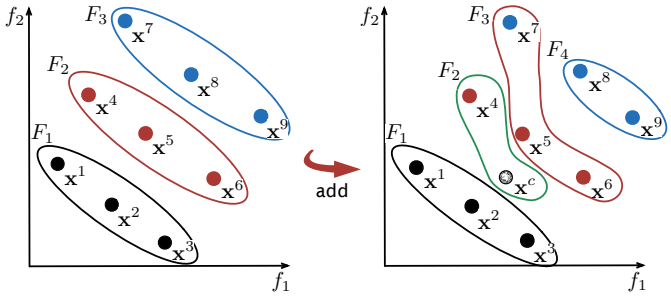


Fig. 6: An example of CASE IV in ENLU method after reproduction.

and $\mathbf{x}' \preceq \mathbf{x}^{\mathsf{E}}$. According to the transitivity property of 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 discussion, Algorithm 3 gives the pseudo-code of ENLU method after environmental selection. To start with, we locate the NDL $F_i$ to which $\mathbf{x}^{\mathsf{E}}$ belongs (line 1 in Algorithm 3). Then, we identify the solutions dominated by $\mathbf{x}^{\mathsf{E}}$. 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 those in $F_i$. The solutions in $S$ and are dominated by those in $F_i$ are stored into a temporary archive $D$ (line 5 in Algorithm 3), whereas those are non-dominated with all 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 $F_{i+1}$ (line 6 to line 8 in Algorithm 3). Note that if $\mathbf{x}^{\mathsf{E}} \in F_l$, 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 non-dominated with $\mathbf{x}^7$, it is added to $F_3$. At last, the ENLU method terminates.

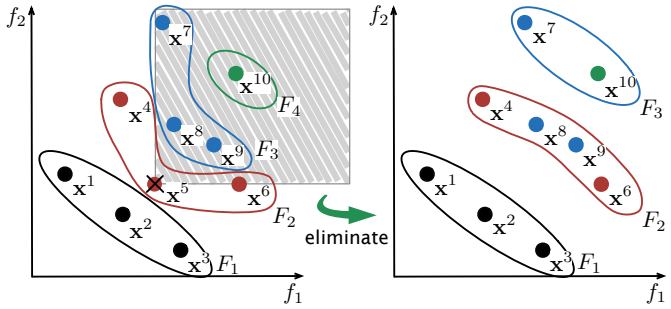


Fig. 7: An example of ENLU method after environmental selection

## 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 (line 3 to line 5 in Algorithm 1).

**Algorithm 3:** ENLU after environmental selection

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

**Output**: updated NDL structure $F$

```
1  Locate the NDL F_i to which x^E belongs;
2  while i < l do
3      S ← solutions in F_{i+1} ∧ dominated by x^E;
4      if S ≠ ∅ then
5          D ← solutions in S ∧ dominated by those in F_i;
6          if D = S then
7              break;
8          end
9          F ← F_i ⋃ S\D;
10     end
11     i++;
12 end
13 return F
```

In the following paragraphs, we consider the computational complexity in two different scenarios, i.e., ENLU method after reproduction and environmental selection, respectively.

### *A. Best-case Complexity of ENLU Method*

Let us first consider the scenario of ENLU method after reproduction. The best-case happens when $F_1$ only contains a single solution and it is non-dominated 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 ENLU method after reproduction is $\mathcal{O}(m)$. As for the scenario of 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 [18] 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 ENLU method after reproduction.

**Lemma 1.** *Given a population having $N$ solutions, which form $l$ ($l \geq 1$) NDLs, i.e., $F_1, \cdots, F_l$. Each $F_i$ contains $\varphi_i$ ($1 \leq \varphi_i \leq N$) solutions, where $i \in \{1, \cdots, l\}$, and $\sum_{i=1}^{l} \varphi_i = N$. The largest number of compairions ($NoC$) is calculated as:*

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

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

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

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

The proof of Lemma 2 can be found in Appendix B. Unfortunately, it is far from trivial to directly derive the NDL structure that maximizes $NoC$ 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 $\varphi_i$, where $i \in \{1, \cdots, l\}$, 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 $l$ varying from 3 to 6 in our experiment for illustrative purpose. Specifically, we have the following results:

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

| $\varphi_1$ | $\varphi_2$ | $\varphi_3$ |
|---|---|---|
| 15 | 14 | 1 |

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

| $\varphi_1$ | $\varphi_2$ | $\varphi_3$ | $\varphi_4$ |
|---|---|---|---|
| 14 | 14 | 1 | 1 |
| 15 | 13 | 1 | 1 |

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

| $\varphi_1$ | $\varphi_2$ | $\varphi_3$ | $\varphi_4$ | $\varphi_5$ |
|---|---|---|---|---|
| 14 | 13 | 1 | 1 | 1 |

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

| $\varphi_1$ | $\varphi_2$ | $\varphi_3$ | $\varphi_4$ | $\varphi_5$ | $\varphi_6$ |
|---|---|---|---|---|---|
| 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 $l \geq 3$, the NDL structure $\varphi_1 = [\frac{N-l+3}{2}]$, $\varphi_2 = N - [\frac{N-l+3}{2}] - 1$, $\varphi_i = 1$, $i \in \{3, \cdots, l\}$ maximizes $NoC$, where $[*]$ can either be a rounded up or rounded down operation in case $\frac{N-l+3}{2}$ is not an integer.*

The proof of Lemma 3 can be found in Appendix C. This lemma provides the theoretical support to the first observation in the above exhaustive search, and it also gives the corresponding NDL structure maximizing $NoC$ 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. 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 ENLU method after reproduction is $\mathcal{O}(mN^2)$. Obviously, the computational complexity of 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 ENLU method with five popular NDS algorithms on two different synthetic data sets. 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 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 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. Non-dominated Sorting Algorithms

*1) Fast Non-dominated Sorting (FNDS) [2]:* Each solution is compared with other solutions in the population, and solutions that are non-dominated with others are assigned to $F_1$. Then, solutions in $F_1$ are temporarily removed from the population, and the remaining non-dominated 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 (DS) [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 (CS) [6]:* Its basic idea is to use the non-dominated 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 non-dominated solutions that are unique for CS.

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

*4) Efficient Non-dominated Sort (ENS) [13]:* In 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 [14], 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. This issue will be further discussed in Section V-E.

### C. Experiments on Cloud Data Set

In this section, we test the performance of ENLU method with 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 5,000 with an increment of 100. That is to say, for a given number of objectives, there are 50 randomly generated populations in total for 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 a 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 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 non-dominated 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, ENLU method shows a constantly best performance in all comparisons. Its superiority becomes even more significant with the increase of the number of objectives. It is interesting to note that the trajectories of ENLU method fluctuate significantly in two- and five-objective cases, and become stable later on. As discussed in Section IV, the computational cost of 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 ENLU method becomes stable in the high-dimensional cases. The issue of NDL structure will be further explored in Section V-E.

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, 1,000, 3,000 and 5,000 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

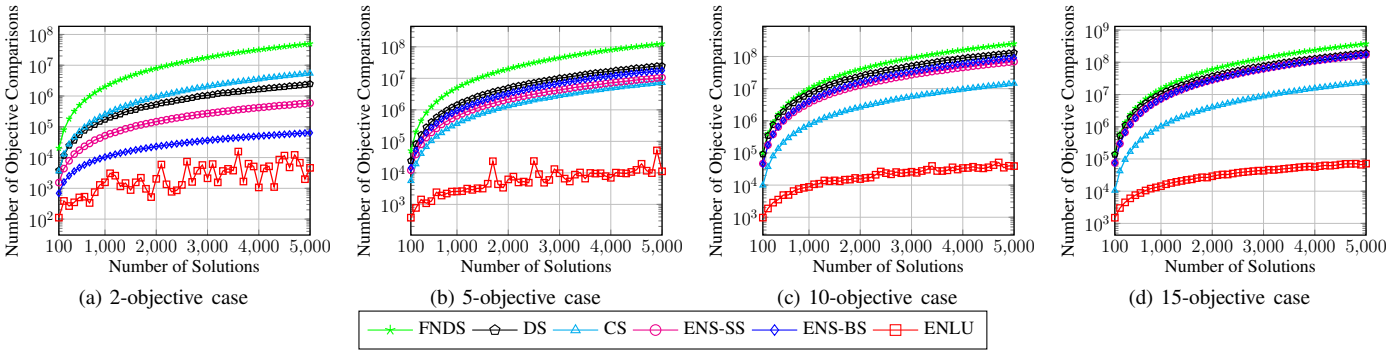


Fig. 8: Median number of objective comparisons of ENLU method and the other five NDS algorithms for cloud data sets.

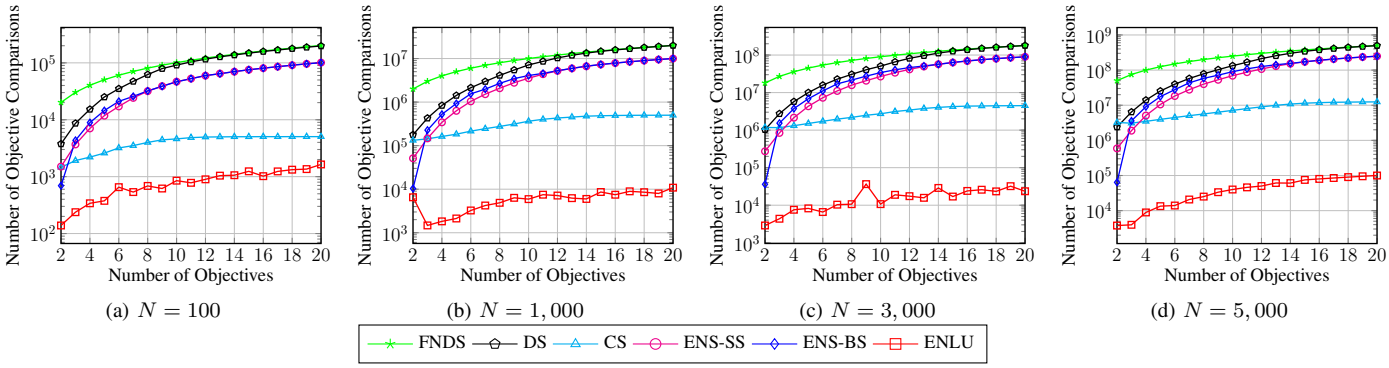


Fig. 9: Median number of objective comparisons of ENLU method and the other five NDS algorithms for cloud data sets with fixed sizes.

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*

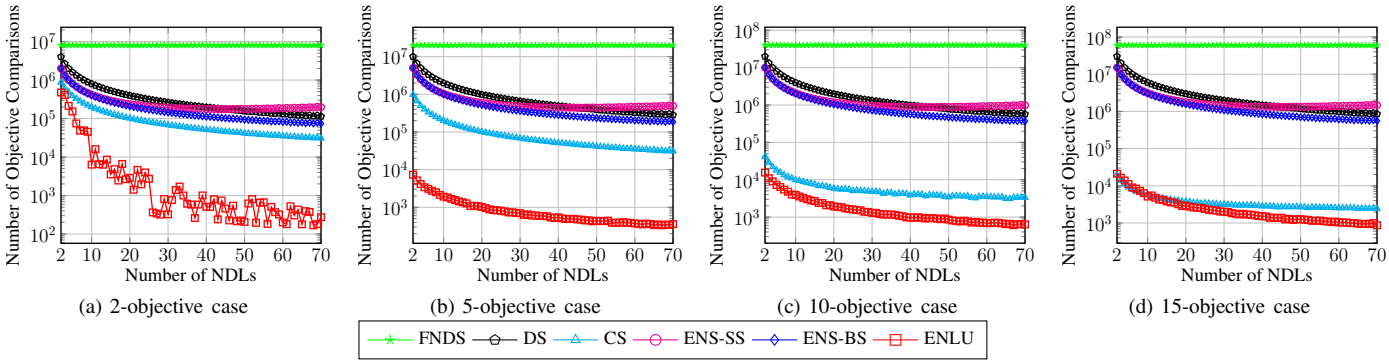


Fig. 10: Median number of objective comparisons of ENLU method and the other five NDS algorithms for fixed fronts data sets with controllable number of NDLs.

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 fixed fronts data sets with two, five, ten and fifteen objectives. The population size is fixed to 2,000, and the number of NDLs varies from 2

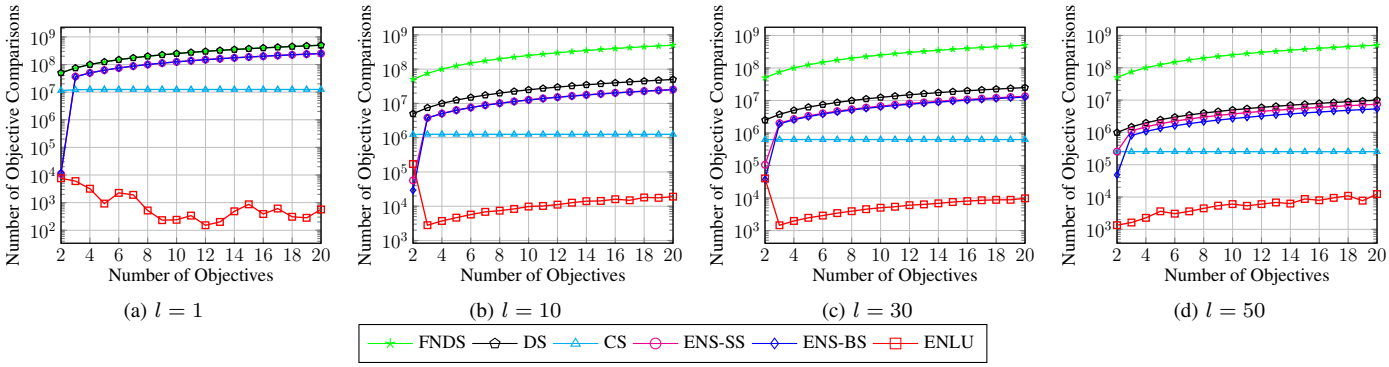


Fig. 11: Median number of objective comparisons of ENLU method and the other five NDS algorithms for fixed fronts data sets with fixed number of NDLs.

to 70 with an increment of one, resulting in 69 populations in total for each test case. Fig. 10 presents the comparison results of ENLU method with FNDS, DS, CS, ENS-SS and ENS-BS, regarding the number of objective comparisons. Similar to the observations for 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 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 5,000, 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 2-objective case, their trajectories surge up towards 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 non-dominated with each other. As for CS, its required number of objective comparisons keep stable all the time.

### *E. Further Investigations of NDL Structure*

In this section, we investigate some interesting properties of the NDL structure. As discussed in [15], for a randomly generated population, the number of NDLs diminishes with the increase of the number of objectives. To validate this assertion, we conduct an experiment on several randomly generated populations. The population size is set as $N = 200$, $N = 800$ and $N = 2,000$, respectively. Each point in a population is randomly sampled from a uniform distribution within the range $\prod_{i=1}^{m}[0, 1]$. The number of objectives varies from 2 to 15 with an increment of one. As shown in Fig. 12, all three trajectories drop rapidly with the growth of dimensionality. In addition, we also notice that the deterioration of the number of NDLs is more significant in case the population size is small.

In real optimization scenarios, decision maker might be more interested in the amount of solutions in the first few NDLs. Here, we conduct another experiment to investigate the variation of the amount of solutions in the first five NDLs for different number of objectives. From the results shown in Fig. 13, we find that the trajectories for different population sizes share a similar trend. Specifically, the amount of solutions in the first NDL steadily increases with the growth of dimensionality, while for the other four NDLs, the amount of solutions slightly increases at first but rapidly decreases later on. Note that this observation conforms to the previous experiment where the number of NDLs decreases with the growth of dimensionality.

The experiments on a randomly generated population demonstrate some interesting properties of NDL structure in the early stage of evolution. It is also interesting to investigate the scenarios in the late stage of evolution, where the population almost approaches the Pareto-optimal front. To this end, we design another synthetic data set where each sample point is within a band region right upon the Pareto-optimal front of DTLZ2 [12]. Fig. 14 presents a simple example of 800 such randomly sampled points in a three-dimensional space. More specifically, the mathematical formulation of DTLZ2 is as follows:

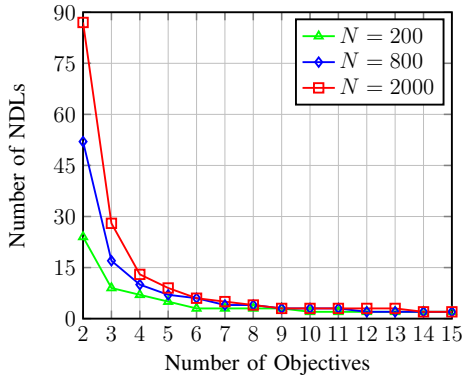


Fig. 12: Number of NDLs formed in a random population, with population size 200, 800 and 2,000, respectively, for different number of objectives.

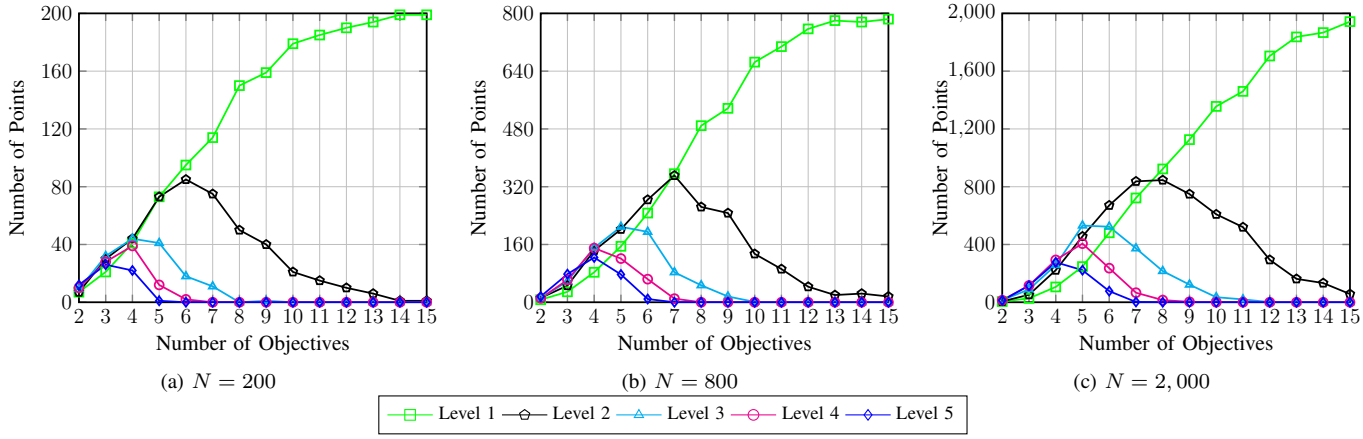


Fig. 13: Number of points in the first five NDLs for different number of objectives.

$$
\begin{aligned}
f_1(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\frac{x_1\pi}{2})\cdots\cos(\frac{x_{m-2}\pi}{2})\cos(\frac{x_{m-1}\pi}{2})\\
f_2(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\frac{x_1\pi}{2})\cdots\cos(\frac{x_{m-2}\pi}{2})\sin(\frac{x_{m-1}\pi}{2})\\
f_3(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\frac{x_1\pi}{2})\cdots\sin(\frac{x_{m-2}\pi}{2})\\
&\vdots\\
f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))\sin(\frac{x_1\pi}{2})
\end{aligned}
$$

where $\mathbf{x} \in \Omega = \prod_{i=1}^{n}[0,1]$ and $g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m}(x_i - 0.5)^2$. To obtain a sample point as shown in Fig. 14, we set $x_i$, where $i \in \{1, 2, \cdots, m-1\}$, be sampled from the range $[0, 1]$ and $x_j$, where $j \in \{m, m+1, \cdots, n\}$, be sampled from the range $[0.5, 0.9]$. Similar to the previous experiments on the randomly generated population, we also investigate two aspects, i.e., the variation of the number of NDLs in different dimensionality and the variation of the amount of solutions in the first five NDLs. From the experimental results shown in Fig. 15 and Fig. 16, we find the trajectories share a similar trend as those in Fig. 12 and Fig. 13. However, it is also worth noting that the number of NDLs formed in this synthetic data set is much fewer than that in the randomly generated population. This observation implies that the number of NDL structure becomes relatively simpler when the population almost converges to the Pareto-optimal front.

### *F. 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 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 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

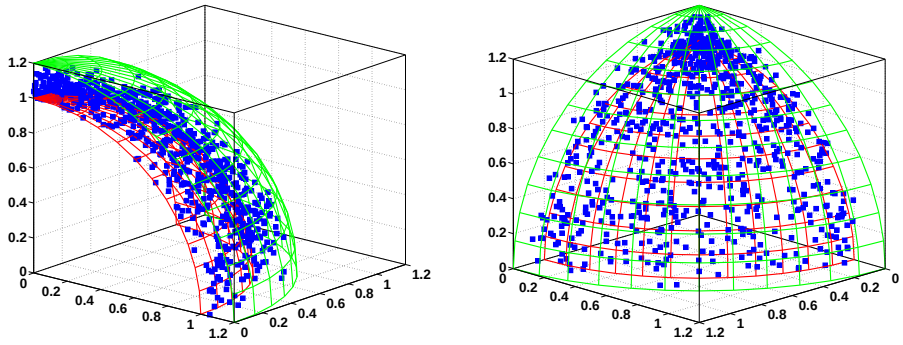

Fig. 14: 800 randomly sampled points within a band region for a three-dimensional case.

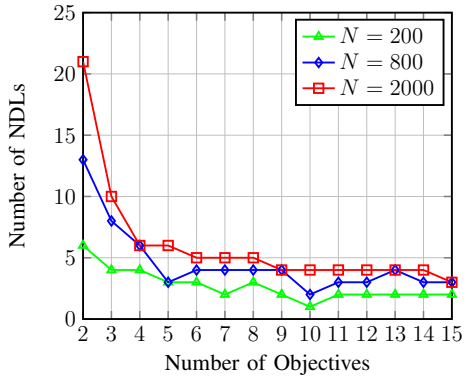


Fig. 15: Number of NDLs formed in the synthetic data sets with population size 200, 800 and 2,000, respectively, for different number of objectives.

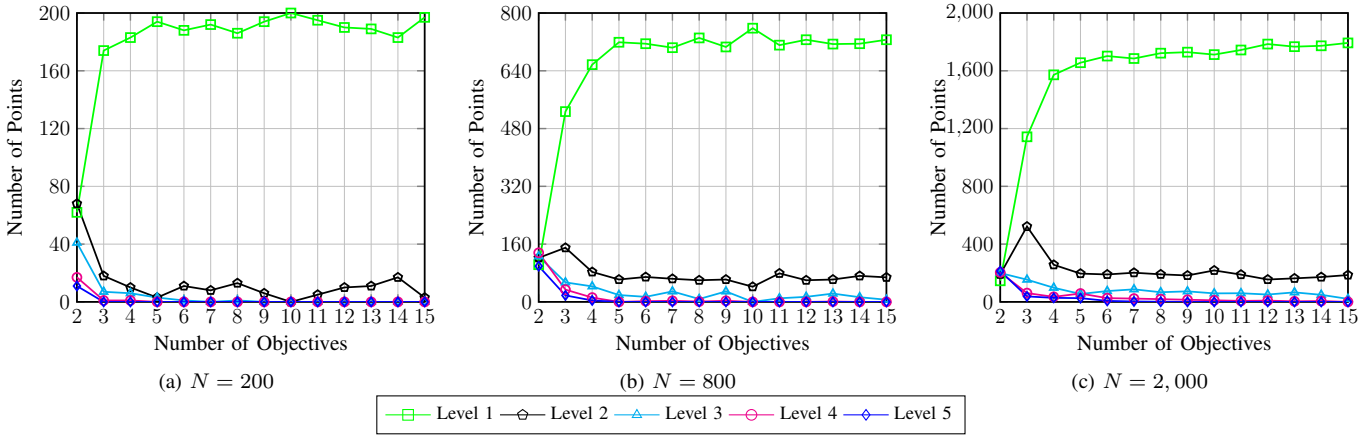


Fig. 16: Number of points in the first five NDLs for different number of objectives.

**Algorithm 4:** Steady-state NSGA-II using ENLU method

**Input**: algorithm parameters
**Output**: population $P$

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

DTLZ4 [12], 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 [16] and polynomial mutation [17] for offspring 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 [18], the settings of the number of generations and population size for different number of objectives are given in Table I.

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 |

Each steady-state NSGA-II variant is launched 21 independent times, and Fig. 17 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 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, 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.

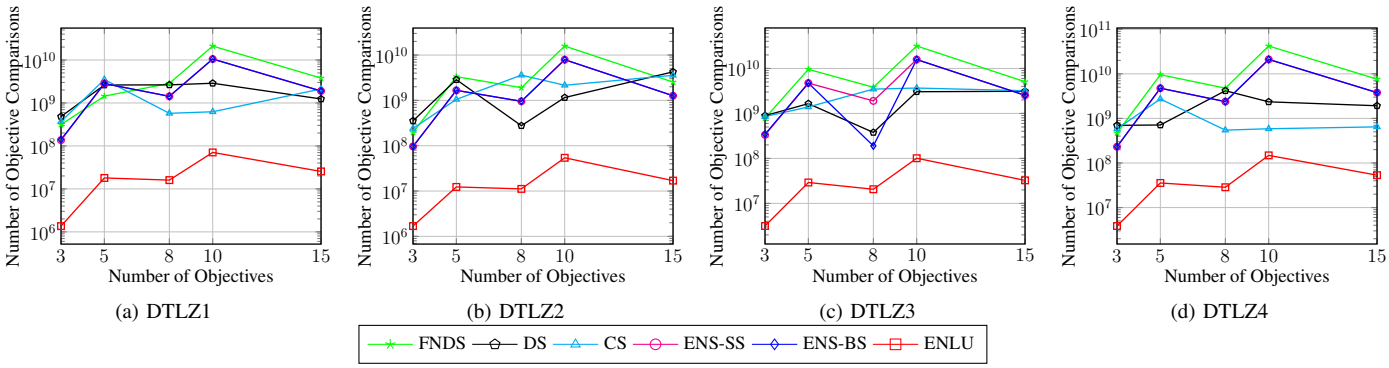


Fig. 17: Median number of objective comparisons of six steady-state NSGA-II variants by using ENLU method and the other five NDS algorithms on DTLZ1 to DTLZ4.

## VI. Conclusions

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 NDS from scratch, this paper presents an ENLU method, which takes advantages of the current population to update the NDL structure, for steady-state EMO. At each iteration, the ENLU method is performed twice: one is after the reproduction, the other is after the environmental selection. By carefully analyzing the population structure, 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 synthetic data sets, but also in real optimization scenarios.

In future, we believe that heuristics (e.g., taking advantage of previous comparisons as done in [5]) and advanced data structure (e.g., K-d tree [19]) are worth being applied to the ENLU method to further improve its computational efficiency. Moreover, the ENLU method proposed in this paper is merely useful for the steady-state evolution model. It is interesting to see whether this idea can be also applicable for the generational evolution model.

## Appendix A
## Proof of Lemma 1

*Proof:* First of all, let us give some terminologies used in this proof.

- $d_i$: The number of solutions dominated by the newly added solutions in $F_i$, where $i \in \{1, \cdots, l\}$ and $0 \leq d_i \leq \varphi_i$.
- indicator function $I_A(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$.
- indicator function $I_B(F_i)$[3].

Let $\tau_i = \varphi_i - d_i - 1$ and $\omega_i = [1 - I_B(F_i)]d_i + I_B(F_i)$, the mathematical model of $NoC$ is formulated as follows:

$$NoC = \varphi_1 + \sum_{i=1}^{l-1}[\prod_{j=1}^{i} I_A(\tau_j)I_A(\omega_j)]\omega_i\varphi_{i+1} \tag{3}$$

According to this model, we can easily find that $NoC$ is constantly equals $\varphi_1$, i.e., $N$, when $l = 1$. When $l \geq 2$, we use induction to finalize our proof.

- When $l = 2$, according to equation (3), we have

$$NoC = \varphi_1 + I_A(\tau_1)I_A(\omega_1)\omega_1\varphi_2 \tag{4}$$

  where $\tau_1 = \varphi_1 - d_1 - 1$ and $\omega_1 = [1 - I_B(F_1)]d_1 + I_B(F_1)$. Obviously, $NoC = \varphi_1$ when the newly added offspring either dominates or is non-dominated with all solutions in $F_1$. Otherwise, in order to maximize $NoC$, we should satisfy the following three conditions:
  1) $I_A(\tau_1) = 1$: it means that $\tau_1 > 0$.
  2) $d_1 > 0$: it means that the newly added offspring dominates some solutions in $F_1$.
  3) $I_B(F_1) = 0$: it means that the newly added offspring are non-dominated with some solutions in $F_1$.

  In this case, $NoC$ is maximized when $d_1 = \varphi_1 - 1$, which means that equation (2) is true for $l = 2$.
- Suppose equation (2) is true for $l = n$, then when $l = n + 1$, we have:

$$NoC = NoC_n + \prod_{j=1}^{n} I_A(\tau_j)I_A(\omega_j)\omega_n\varphi_{n+1} \tag{5}$$

  where $NoC_n = \varphi_1 + \sum_{i=1}^{k}(\varphi_{i-1} - 1)\varphi_i$. $k = n$ in case there does not exist any NDL, before $F_n$, in which the newly added solutions from the previous NDL dominates or are non-dominated with all solutions; otherwise $k$ is the index of the first such NDL. Accordingly, we will have $NoC = NoC_n$ in case there exists a NDL, before $F_l$, in which the newly added solutions from the previous NDL dominates or are non-dominated with all solutions. This is because the production term in equation (5) will be zero in that case. On the flip side, similar to the proof when $l = 2$, we also need to satisfy the following three conditions:
  1) $I_A(\tau_n) = 1$: it means that $\tau_n > 0$.
  2) $d_n > 0$: it means that the newly added solutions, moved from $F_{n-1}$ to $F_n$, dominates some solutions in $F_n$.
  3) $I_B(F_n) = 0$: it means that the newly added solutions, moved from $F_{n-1}$ to $F_n$, are non-dominated with some solutions in $F_n$.

In this case, we can maximize $NoC$ when $d_n = \varphi_n - 1$. All in all, equation (2) holds for $l = n + 1$, and the proof of the induction step is completed. ■

[3] $I_B(F_i) = \begin{cases} 0 & \text{if added solutions are non-dominated with those in } F_i. \\ 1 & \text{otherwise} \end{cases}$

## APPENDIX B
## PROOF OF LEMMA 2

*Proof:* According to Lemma 1, when $l = 2$ the largest $NoC$ is calculated as:

$$NoC = \varphi_1 + (\varphi_1 - 1)\varphi_2 \tag{6}$$

Since $\varphi_1 + \varphi_2 = N$, we use $\varphi_2 = N - \varphi_1$ to do substitution in equation (6):

$$NoC = -\varphi_1^2 + (N+2)\varphi_1 - N \tag{7}$$

To find the value of $\varphi_1$ that maximizes $NoC$, we take the derivative of $\varphi_1$ on both sides of equation (7). This process gives us $\varphi_1 = [\frac{N}{2}] + 1$ and $\varphi_2 = N - [\frac{N}{2}] - 1$. Note that $[*]$ can either be a rounded up or rounded down operation in case $\frac{N}{2}$ is not an integer. ■

## APPENDIX C
## PROOF OF LEMMA 3

*Proof:* According to Lemma 1, we can unfold the equation (2) and rewrite it as follows:

$$\begin{aligned} NoC &= \varphi_1 + \sum_{i=2}^{k}(\varphi_{i-1}\varphi_i - \varphi_i) \\ &= \varphi_1(\varphi_2 + 1) + \sum_{i=2}^{k-1}\varphi_i(\varphi_{i+1} - 1) - \varphi_k \end{aligned} \tag{8}$$

where $\varphi_i \geq 1$, $i \in \{1, \cdots, k\}$, $\sum_{i=1}^{l}\varphi_i = N$ and $l \geq k$. Note that there is only one subtraction term, i.e., $-\varphi_k$, in equation (8). Assume that $\varphi_i$, $i \in \{1, \cdots, k\}$, are unknown to us. If we decrease $\varphi_k$ by 1 and increase $\varphi_{k-2}$ by 1 accordingly, which is used to meet the constraint $\sum_{i=1}^{l}\varphi_i = N$, the resulting $NoC'$ is calculated as:

$$\begin{aligned} NoC' = \varphi_1(\varphi_2 + 1) &+ \sum_{i=2}^{k-4}\varphi_i(\varphi_{i+1} - 1) + \varphi_{k-3}(\varphi'_{k-2} - 1) \\ &+ \varphi'_{k-2}(\varphi_{k-1} - 1) + \varphi_{k-1}(\varphi'_k - 1) - \varphi'_k \end{aligned} \tag{9}$$

where $\varphi'_{k-2} = \varphi_{k-2} + 1$ and $\varphi'_k = \varphi_k - 1$. Then, equation (9) can be further written as:

$$\begin{aligned} NoC' = \varphi_1(\varphi_2 + 1) &+ \sum_{i=2}^{k-4}\varphi_i(\varphi_{i+1} - 1) + \varphi_{k-3}(\varphi_{k-2} - 1) \\ &+ \varphi_{k-3} + \varphi_{k-2}(\varphi_{k-1} - 1) + \varphi_{k-1}(\varphi_k - 1) - \varphi_k \end{aligned} \tag{10}$$

Obvioulsy, we have $NoC' - NoC = \varphi_{k-3} \geq 1$. If this above procedure iterates until $\varphi_k$ is decreased to 1, the resulting $NoC$ will be increased by $(\varphi_k - 1)\varphi_{k-3}$ at the end. Let us rewrite equation (8) as:

$$NoC = \varphi_1(\varphi_2 + 1) + \sum_{i=2}^{k-2}\varphi_i(\varphi_{i+1} - 1) - 1 \tag{11}$$

where $\varphi_i \geq 1$, $i \in \{1, \cdots, k\}$, $\varphi_k = 1$, $\sum_{i=1}^{l}\varphi_i = N$ and $l \geq k$. If we decrease $\varphi_{k-1}$ by 1 and increase $\varphi_{k-3}$ by 1 accordingly, which is used to meet the constraint $\sum_{i=1}^{l}\varphi_i = N$, the resulting $NoC'$ is calculated as:

$$\begin{aligned} NoC' = \varphi_1(\varphi_2 + 1) &+ \sum_{i=2}^{k-5}\varphi_i(\varphi_{i+1} - 1) + \varphi_{k-4}(\varphi'_{k-3} - 1) \\ &+ \varphi'_{k-3}(\varphi_{k-2} - 1) + \varphi_{k-2}(\varphi'_{k-1} - 1) - 1 \end{aligned} \tag{12}$$

where $\varphi'_{k-3} = \varphi_{k-3} + 1$ and $\varphi'_{k-1} = \varphi_{k-1} - 1$. Then, equation (12) can be further written as:

$$\begin{aligned} NoC' = \varphi_1(\varphi_2 + 1) &+ \sum_{i=2}^{k-5}\varphi_i(\varphi_{i+1} - 1) + \varphi_{k-4}(\varphi_{k-3} - 1) \\ &+ \varphi_{k-3}(\varphi_{k-2} - 1) + \varphi_{k-2}(\varphi_{k-1} - 1) - 1 + (\varphi_{k-4} - 1) \end{aligned} \tag{13}$$

Note that $\varphi_{k-4} \neq 1$, otherwise there is no further comparison after $F_{k-4}$ according to Lemma 1. Therefore, according to equation (12) and equation (13), we have $NoC' - NoC = \varphi_{k-4} - 1 > 0$. Similar to the previous derivation, if this above procedure iterates until $\varphi_{k-1}$ is decreased to 1, the resulting $NoC$ will be increased by $(\varphi_{k-1} - 1)(\varphi_{k-4} - 1)$ at the end.

In fact, if we repeat the above derivation, we can have:

$$NoC = \varphi_1(\varphi_2 + 1) - 1 \tag{14}$$

where $\varphi_1 > 1$, $\varphi_2 > 1$, $\varphi_i = 1$, $i \in \{3, \cdots, k\}$, $\sum_{i=1}^{l} \varphi_i = N$ and $l \geq k$. Using $\varphi_2 = N - l + 2 - \varphi_1$ to do substitution in equation (14):

$$NoC = -\varphi_1^2 + (N - l + 3)\varphi_1 - 1 \tag{15}$$

Similar to the proof in Appendix B, we take the derivative of $\varphi_1$ on both side of equation (15) to find the value of $\varphi_1$ that maximizes $NoC$. This process gives us $\varphi_1 = [\frac{N-l+3}{2}]$, $\varphi_2 = N - [\frac{N-l+3}{2}] - 1$, where $[*]$ can either be a rounded up or rounded down operation in case $\frac{N-l+3}{2}$ is not an integer. ■

## APPENDIX D
## PROOF OF THEOREM 1

*Proof:* Lemma 1 and Lemma 3, for a given $N$ and some $l \geq 3$, we have:

$$NoC = \varphi_1 + \varphi_1\varphi_2 - 1 \tag{16}$$

where $\varphi_1 = [\frac{N-l+3}{2}]$, $\varphi_2 = N - [\frac{N-l+3}{2}] - 1$ and $\varphi_i = 1$, $i \in \{3, \cdots, l\}$. Obviously, in order to maximize $NoC$ in equation (16), we need to make $l = 3$ so that $\varphi_1$ and $\varphi_2$ do not need to sacrifice points to the other residual $\varphi_i$s. In other words, for a given $N$, $l = 3$ (when $l \geq 3$) maximizes $NoC$ and the corresponding NDL structure is $\varphi_1 = [\frac{N}{2}]$, $\varphi_2 = N - [\frac{N}{2}] - 1$ and $\varphi_3 = 1$, where $[*]$ can either be a rounded up or rounded down operation in case $\frac{N}{2}$ is not an integer. Using these $\varphi_i$s, $i \in \{1, 2, 3\}$, in equation (16), we have the maximum $NoC$, denoted as $NoC_{max}$, is calculated as:

$$NoC_{max} = -[\frac{N}{2}]^2 + [\frac{N}{2}]N - 1 \tag{17}$$

On the other hand, according to Lemma 1 and Lemma 2, we have the maximum $NoC$ when $l = 2$, denoted as $NoC'_{max}$, is calculated as:

$$NoC'_{max} = -[\frac{N}{2}]^2 + [\frac{N}{2}]N + 1 \tag{18}$$

Obviously, $NoC'_{max} > NoC_{max}$. ■

## REFERENCES


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

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 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," *Journal of the ACM*, vol. 22, no. 6, pp. 469–476, Oct. 1975.

[4] 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.

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

[6] 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.

[7] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

[8] 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 *GECCO'15: Proc. of the 2015 Genetic and Evolutionary Computation Conference*, 2015, pp. 647–654.

[9] 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 *EMO'09: Proc. of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, 2009, pp. 183–197.

[10] 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*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 193, pp. 435–456.

[11] M. Buzdalov and V. Parfenov, "Various degrees of steadiness in NSGA-II and their influence on the quality of results," in *GECCO'15: Proc. of the 2015 Genetic and Evolutionary Computation Conference*, 2015, pp. 749–750.

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

[13] S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 442–446, 2012.

[14] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[15] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *CEC'08: Proc. of the 2008 IEEE Congress on Evolutionary Computation*, 2008, pp. 2419–2426.

[16] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 1–34, 1994.

[17] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

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

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