Time Complexity Analysis of the Deductive Sort in the Best Case

Sumit Mishra
IIIT Guwahati
Guwahati, India
sumit@iiitg.ac.in

Ved Prakash
IIIT Guwahati
Guwahati, India
johnvedprakash@gmail.com

ABSTRACT

Non-dominated sorting is one of the important step in multiobjective evolutionary algorithms (MOEAs) which are based on Pareto dominance concept. Though non-dominated sorting can be performed in polynomial time, it remains an asymptotical bottleneck in many of these MOEAs. Here we show that an algorithm, Deductive Sort from the paper "Deductive Sort and Climbing Sort: New Methods for Non-Dominated Sorting" by McClymont *et al.*, has the best-case time complexity of $\Theta(MN\sqrt{N} + N^2)$.

CCS CONCEPTS

• **Theory of computation → Sorting and searching**; • **Mathematics of computing →** Mathematical optimization;

KEYWORDS

Non-dominated sorting, dominance comparisons, time complexity

ACM Reference Format:
Sumit Mishra and Ved Prakash. 2021. Time Complexity Analysis of the Deductive Sort in the Best Case. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459416

Algorithm 1 Deductive Sort

```
Require: $\mathbb{P} = \{s_1, s_2, \dots, s_N\}$: points in $M$-dimensional space
Ensure: $\mathbb{F} = \{F_1, F_2, \dots\}$: points from $\mathbb{P}$ split into fronts
 1: $x \leftarrow 1$                                          ▷ Set first front
 2: $f \leftarrow 0$                        ▷ Number of sorted solutions to be 0
 3: isSorted[1...N] ← False                     ▷ Boolean sorted flag array
 4: $\mathbb{F} \leftarrow \emptyset$                            ▷ Initialize set of fronts
 5: while $f < N$ do                      ▷ While not all solutions are sorted
 6:     $F_x \leftarrow \emptyset$                                 ▷ Initialize a front
 7:     isDominated[1...N] ← False           ▷ Boolean dominated flag array
 8:     for $i \leftarrow 1$ to $N$ do             ▷ Iterate through solutions in $\mathbb{P}$
 9:         if !isDominated[i] & !isSorted[i] then
10:             for $j \leftarrow i+1$ to $N$ do        ▷ From next solution to last
11:                 if !isDominated[j] & !isSorted[j] then
12:                     domRel ← Dominates($s_i$, $s_j$)   ▷ Obtain dominance
                        relationship between $s_i$ and $s_j$
13:                     if domRel = 1 then                  ▷ $s_i$ dominates $s_j$
14:                         isDominated[j] ← True
15:                     else if domRel = 3 then             ▷ $s_j$ dominates $s_i$
16:                         isDominated[i] ← True
17:                         Break
18:             if !isDominated[i] then                 ▷ $s_i$ is not dominated
19:                 $F_x \leftarrow F_x \cup \{s_i\}$                    ▷ Insert $s_i$ into $F_x$
20:                 isSorted[i] ← True
21:                 $f \leftarrow f + 1$                      ▷ Increment number sorted
22:     $\mathbb{F} \leftarrow \mathbb{F} \cup \{F_x\}$                            ▷ Add $F_x$ to $\mathbb{F}$
23:     $x \leftarrow x + 1$                             ▷ Increment current front
24: Return $\mathbb{F}$                                 ▷ Return the set of fronts
```

1 INTRODUCTION

Modern multiobjective evolutionary algorithms (MOEAs) often rank the solutions in the population based on the Pareto dominance relation. Let $\mathbb{P} = \{s_1, s_2, \dots, s_N\}$ be a population of size N in $\mathbb{R}^M$ where each solution $s_i, 1 \leq i \leq N$ is represented as follows $s_i = \langle s_{i_1}, s_{i_2}, \dots, s_{i_M}\rangle$. Without loss of generality, we assume minimization of each of M objectives, and say that a solution s_i *dominates* a solution s_j, denoted as $s_i \prec s_j$, if $\forall m$ it holds that $s_{i_m} \leq s_{j_m}$, and $\exists m$ such that $s_{i_m} < s_{j_m}$.

Non-dominated sorting divides the population $\mathbb{P}$ into disjoint fronts $F_1, F_2, \dots$ such that all the solutions in a particular front does not dominate each other and for every solution $s \in F_{i+1}$ there exists a solution $s' \in F_i$ such that $s' \prec s$. Such a partition is unique. If a solution belongs to a front F_i, it is said to have a *rank i*.

Fast non-dominated sorting algorithm [1] made the non-dominated sorting procedure popular, and after that, many researchers have started working towards improving the efficiency of the non-dominated sorting procedure. Among those approaches, we consider the algorithm called the Deductive Sort [2]. The original paper claimed that the best-case occurs when N solutions are equally divided into $\sqrt{N}$ fronts such that each solution in a front is dominated by all the solutions in its preceding front. It is assumed that the first solution selected in each iteration is in the current front. In this case, the number of dominance comparisons is $\frac{N\sqrt{N}-1}{2} + \frac{\sqrt{N}(\sqrt{N}-1)}{2}$ [2]. Recently, the worst-case time complexity of Deductive Sort is analyzed in [3]. Similarly, the worst-case time complexity DDA-NS [7] is analyzed in [4]. In this paper, we show that the number of dominance comparisons in the best case by Deductive Sort is $N(\sqrt{N}-1)$ (which is approx. double than the claimed one), and the time complexity is $O(MN\sqrt{N} + N^2)$.

2 ALGORITHM

Initially, all the solutions are *unranked* and not *dominated.* In each iteration, a pivot solution is selected from the population (which is neither *ranked* nor *dominated*) and is compared with only *unranked* and not *dominated* solutions. The solutions which are dominated by the pivot is marked *dominated.* If the pivot is dominated, then we choose another pivot and again start comparing the pivot with other solutions. At the end of the iteration, all the non-dominated solutions are comprised of the desired front. The Deductive Sort algorithm is summarized in Algorithm 1.

3 ANALYSIS

In the best case, N solutions are equally divided into $\sqrt{N}$ fronts. Let the solutions in front F_1 be $\{s_1, s_2, \ldots, s_{\sqrt{N}}\}$, the solutions in front F_2 be $\{s_{\sqrt{N}+1}, s_{\sqrt{N}+2}, \ldots, s_{2\sqrt{N}}\}$, the solutions in front F_3 be $\{s_{2\sqrt{N}+1}, s_{2\sqrt{N}+2}, \ldots, s_{3\sqrt{N}}\}$ and so on. In general, the solutions in front F_K be $\{s_{(K-1)\sqrt{N}+1}, s_{(K-1)\sqrt{N}+2}, \ldots, s_{K\sqrt{N}}\}$. For the best case, the first solution selected in each iteration should be in the current front so the ordering of the solutions should be: $\{F_1, F_2, \ldots, F_{\sqrt{N}}\}$. However, among the $\sqrt{N}$ solutions of a particular front, any ordering will be suffice.

In each iteration (*While Loop*) of the algorithm, the array `isDominated[ ]` is initialized. Initially all the N solutions are *unsorted* (line 3). The number of unsorted (unranked) solutions after obtaining the solutions of $F_1, F_2, \ldots, F_{K-1}$ (or before obtaining the front F_K) is $N - (K-1)\sqrt{N}$. While obtaining F_K, the first solution of front F_K ($s_{(K-1)\sqrt{N}+1}$) is compared with all the remaining unranked solutions. Thus, the number of solutions to which $s_{(K-1)\sqrt{N}+1}$ is compared = $\left(N - (K-1)\sqrt{N} - 1\right)$. During this comparison, all the solutions of $F_{K+1}, F_{K+2}, \ldots, F_{\sqrt{N}}$ are marked *dominated*. Only the solutions of F_K are not marked *dominated*. Now the second solution of F_K ($s_{(K-1)\sqrt{N}+2}$) is compared with $s_{(K-1)\sqrt{N}+3}, s_{(K-1)\sqrt{N}+4}, \ldots, s_{K\sqrt{N}}$ so the second solution is compared with $\sqrt{N} - 2$ solutions. Similarly, the third solution of F_K ($s_{(K-1)\sqrt{N}+3}$) is compared with $s_{(K-1)\sqrt{N}+4}, s_{(K-1)\sqrt{N}+5}, \ldots, s_{K\sqrt{N}}$ so the third solution is compared with $\sqrt{N} - 3$ solutions and so on. Thus, while obtaining front F_K, the number of dominance comparisons by the Deductive Sort in the best case is obtained by Eq. (1).

$$\begin{aligned}\mathtt{DC}_K &= \left(N - (K-1)\sqrt{N} - 1\right) + \left(\sqrt{N} - 2\right) + \left(\sqrt{N} - 3\right) + \ldots + 1 \\ &= \left(N - (K-1)\sqrt{N} - 1\right) + \frac{1}{2}\left(\sqrt{N} - 2\right)\left(\sqrt{N} - 1\right) \qquad (1)\end{aligned}$$

Thus, the total number of dominance comparisons by the Deductive Sort in the best case is obtained by Eq. (2).

$$\mathtt{DC} = \sum_{K=1}^{\sqrt{N}} \mathtt{DC}_K = N\left(\sqrt{N} - 1\right) \qquad (2)$$

However, the authors claimed that the number of dominance comparisons in this case is $\frac{N\sqrt{N}-1}{2} + \frac{\sqrt{N}(\sqrt{N}-1)}{2}$ [2]. The ratio of the claimed number of dominance comparisons to the actual number of dominance comparisons is obtained by Eq. (3).

$$\begin{aligned}R &= \lim_{N\to\infty} \frac{\mathtt{Claimed}}{\mathtt{Obtained}} \\ &= \lim_{N\to\infty} \frac{\frac{N\sqrt{N}-1}{2} + \frac{\sqrt{N}(\sqrt{N}-1)}{2}}{N\left(\sqrt{N}-1\right)} \\ &= \lim_{N\to\infty} \frac{N\sqrt{N} + N - \sqrt{N} - 1}{2N\sqrt{N} - 2N} = \frac{1}{2} \qquad (3)\end{aligned}$$

Thus, the claimed number of dominance comparisons is approximately half the correct number of dominance comparisons.

The number of unsorted (unranked) solutions after obtaining the solutions of $F_1, F_2, \ldots, F_{K-1}$ (or before obtaining the front F_K) is $N - (K-1)\sqrt{N}$. For the i^{th} solution (say s_i), *While* loop in line 10 runs $N - i + 1$ times. However, it is not necessary that solution s_i is compared with $N - i + 1$ solutions. It depends whether a solution has been already *ranked* or it has been marked *dominated* by some other solution. Thus while obtaining F_K, for solution $s_{(K-1)\sqrt{N}+1}$ line 10 runs $N - (K-1)\sqrt{N} - 1$ times. For solution $s_{(K-1)\sqrt{N}+2}$ line 10 runs $N - (K-1)\sqrt{N} - 2$ times and so on. For solution $s_{K\sqrt{N}}$ line 10 runs $N - (K-1)\sqrt{N} - \sqrt{N}$ times. Thus, to obtain the K^{th} front, the number of times line 10 executes is obtained by Eq. (4).

$$\begin{aligned}\mathtt{C}_K &= \left[N - (K-1)\sqrt{N} - 1\right] + \left[N - (K-1)\sqrt{N} - 2\right] + \\ &\quad \cdots + \left[N - (K-1)\sqrt{N} - \sqrt{N}\right] \\ &= \frac{\sqrt{N}}{2}\left[2N - (2K-1)\sqrt{N} - 1\right] \qquad (4)\end{aligned}$$

Thus, the total number of times line 10 executes in the Deductive Sort in the best case is obtained by Eq. (5).

$$\mathtt{C} = \sum_{K=1}^{\sqrt{N}} \mathtt{C}_K = \frac{1}{2}N(N-1) = O(N^2) \qquad (5)$$

Thus, the time complexity of Deductive Sort in the best case is $\Theta(MN\sqrt{N} + N^2)$.

4 CONCLUSION & FUTURE WORK

In this paper, we have shown that the number of dominance comparisons by the Deductive Sort in its best-case is $N(\sqrt{N} - 1)$. We have also proved that the best-case time complexity of Deductive Sort is $\Theta(MN\sqrt{N} + N^2)$. The worst-case time complexity of Deductive Sort is proved to be $\Theta(MN^3)$ in [3]. This means this algorithm is not useful in performance-critical systems (in-spite of being simple), especially those which are outside the domain of evolutionary computation. For relatively small population size, this sorting can be a good choice as it does not presort the solutions, unlike various recent approaches [5, 6] and also it only uses the array data structure.

REFERENCES

[1] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197.

[2] Kent McClymont and Ed Keedwell. 2012. Deductive Sort and Climbing Sort: New Methods for Non-Dominated Sorting. *Evolutionary Computation* 20, 1 (Spring 2012), 1–26.

[3] Sumit Mishra and Maxim Buzdalov. 2020. If Unsure, Shuffle: Deductive Sort is $\Theta(MN^3)$, but $O(MN^2)$ in Expectation over Input Permutations. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'2020).* 516–523.

[4] Sumit Mishra, Maxim Buzdalov, and Rakesh Senwar. 2020. Time Complexity Analysis of the Dominance Degree Approach for Non-Dominated Sorting. In *Proceedings of Genetic and Evolutionary Computation Conference Companion (GECCO'2020).* 169–170.

[5] Sumit Mishra, Sriparna Saha, Samrat Mondal, and Carlos A. Coello Coello. 2019. A Divide-And-Conquer Based Efficient Non-Dominated Sorting Approach. *Swarm and Evolutionary Computation* 44 (February 2019), 748–773.

[6] Proteek Chandan Roy, Kalyanmoy Deb, and Md Monirul Islam. 2018. An Efficient Nondominated Sorting Algorithm for Large Number of Fronts. *IEEE Transactions on Cybernetics* 99 (2018), 1–11.

[7] Yuren Zhou, Zefeng Chen, and Jun Zhang. 2017. Ranking Vectors by Means of the Dominance Degree Matrix. *IEEE Transactions on Evolutionary Computation* 21, 1 (2017), 34–51.