Labeling-Oriented Non-Dominated Sorting is $\Theta(MN^3)$

Sumit Mishra
IIIT Guwahati
Assam, India

Ved Prakash
IIIT Guwahati
Assam, India

Maxim Buzdalov
ITMO University
Saint Petersburg, Russia

ABSTRACT

Non-dominated sorting is a common routine used in many evolutionary multiobjective algorithms to rank solutions based on the Pareto-dominance relation. Unlike many other problems appearing in evolutionary computation, this problem has a simple formal definition, a number of quite efficient algorithms to solve it, and is relatively well understood.

Unfortunately, a number of recent papers that propose new, and supposedly efficient, algorithms for non-dominated sorting, feature inaccurate analysis, leading to overly optimistic claims. In this paper we prove that a recent algorithm, called Labeling-Oriented Non-Dominated Sorting, or LONSA, has the worst-case running time of $\Theta(MN^3)$, where N is the number of points and M is the number of objectives, which is much greater than the quadratic upper bound the authors claim. Our proof holds for all $M \geq 4$ and essentially reduces LONSA to another algorithm, Deductive Sort, for which the hard test has been constructed before.

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, Ved Prakash, and Maxim Buzdalov. 2021. Labeling-Oriented Non-Dominated Sorting is $\Theta(MN^3)$. 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/10.1145/3449726.3459425

1 INTRODUCTION

In multiobjective optimization, the Pareto dominance relation, or simply dominance, is defined as follows: a point p dominates a point q, written as $p \prec q$, if the two following conditions are satisfied:

- $\forall m \in \{1, 2, \ldots, M\}\ p_m \leq q_m$;
- $\exists m \in \{1, 2, \ldots, M\}\ p_m < q_m$;

where M is the number of objectives, and it is assumed that each of the objectives needs to be minimized. If neither $p \prec q$ nor $q \prec p$, the points p and q are said to be non-dominated with each other.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459425

Algorithm 1 LONSA

Require: $\mathbb{P} = \{s_1, s_2, \ldots, s_N\}$: points in M-dimensional space
Ensure: $\mathbb{F} = \{F_1, F_2, \ldots\}$: points from $\mathbb{P}$ split into fronts
1: $\texttt{array}_x \leftarrow$ Sort the population using the objective 1
2: $\texttt{array}_y \leftarrow$ Sort the population using the objective 2
3: $\tau \leftarrow 1$
4: **while** $\mathbb{P} \neq \emptyset$ **do**
5: $\quad F_\tau \leftarrow$ LONSA-MANY($\mathbb{P}, \texttt{array}_x, \texttt{array}_y$)
6: $\quad \mathbb{F} \leftarrow \mathbb{F} \cup F_\tau$
7: $\quad \tau \leftarrow \tau + 1$
8: **return** $\mathbb{F}$

Let $\mathbb{P} = \{p_1, p_2, \ldots, p_N\}$ be a population of N points, each of dimension M. Non-dominated sorting is a procedure that produces a set of *fronts* $\mathbb{F} = \{F_1, F_2, \ldots\}$ such that the following holds:

- The union of all fronts is the whole population: $\bigcup_{k \geq 1} F_k = \mathbb{P}$;
- The fronts are disjoint: $\forall i \neq j\ F_i \cap F_j = \emptyset$;
- All the points in a particular front are non-dominated with each other: $\forall k \geq 1\ \forall p, q \in F_k : p \not\prec q, q \not\prec p$;
- The first front consists of points that are not dominated by any point: $\forall q \in F_1\ \nexists p \in \mathbb{P} : p \prec q$;
- All the points in any front except for the first one are dominated by at least one of the points in the preceding front: $\forall q \in F_k, k > 1\ \exists p \in F_{k-1} : p \prec q$.

Non-dominated sorting was proposed for use in multiobjective optimization as a part of NSGA [5] with an algorithm of time complexity $\Theta(MN^3)$, and made popular in the paper that proposed NSGA-II [2] along with an algorithm of time complexity $\Theta(MN^2)$. Since then, many different algorithms with various efficiency have been proposed to solve this problem. One of the reasons is that non-dominated sorting appears to have no trivial and efficient algorithm, and even its theoretical computational complexity is not yet well-understood [6].

In this paper we consider the algorithm titled "Labeling-Oriented Non-Dominated Sorting Algorithm", or LONSA, proposed in [1]. Its basic idea is to first perform pre-sorting of the points in the first two objectives and store the points in these orders in two arrays, $\texttt{array}_x$ and $\texttt{array}_y$, which are then used to accelerate the pretty standard procedure of peeling the fronts one by one, beginning with the first one, similar to what the first proposed algorithm in [5] does. The top-level algorithm is given in Algorithm 1, and the front peeling procedure, which is used when $M > 2$, is given in Algorithm 2.

The basic idea of Algorithm 2 is to maintain a label on each point, which can take one of three values: #1 means the point has not been yet evaluated, #2 means it probably belongs to the current front being peeled off, and #3 means it was dominated by one of the remaining points and hence does not belong to the current

Algorithm 2 LONSA-MANY($\mathbb{P}$, $\texttt{array}_x$, $\texttt{array}_y$)

Require: $\mathbb{P} = \{s_1, s_2, \ldots, s_N\}$: points in M-dimensional space
Ensure: $\mathbb{F} = \{F_1, F_2, \ldots\}$: points from $\mathbb{P}$ split into fronts

```
for each s ∈ array_x do
    label(s) ← #1
for each s ∈ array_x do
    if label(s) = #1 then
        label(s) ← #2
        pos_y ← FIND_y(s) ▷ Obtain the position of s in array_y
        while pos_y < |array_y| do
            if label(array_y[pos_y]) ≠ #3 then
                if s ≺ array_y[pos_y] then
                    label(array_y[pos_y]) ← #3
                else if array_y[pos_y] ≺ s then
                    label(array_y[pos_y]) ← #3
                    break
            pos_y ← pos_y + 1
F ← ∅
for each s ∈ array_x do
    if label(s) = #2 then
        ℙ ← ℙ \ {s}
        F ← F ∪ {s}
        Remove s from array_x
        Remove s from array_y
return F
```

front. The points are checked one by one in the order specified by $\texttt{array}_x$. Once a point s with a label #1 is found, its index $\texttt{pos}_y$ is found in $\texttt{array}_y$, and only the points that follow s in that array are checked for dominance with s. Points dominated by s are labeled as #3 and effectively ignored on the current iteration. If s is itself found to be dominated, it is also labeled as #3, but in this case the loop ends, and the process continues with the next point. If the point s survived for the whole loop, it is marked #2. It has chances to enter the current front now, but in fact, as some points have not yet been compared with s at this time, but may do in the future, it can be labeled #3 later.

Once all points have been processed, those labeled #2 constitute the current front, hence they are removed from $\texttt{array}_x$ and $\texttt{array}_y$.

2 ANALYSIS

First of all, we have to note that our presentation of this algorithm assumes, for clarity and efficiency, that the labels are stored together with the other point data, and not along the arrays $\texttt{array}_x$ and $\texttt{array}_y$, as the original pseudocode from [1] can be interpreted. This, in particular, avoids certain lookup procedures characteristic to the original pseudocode. It is also not clear how [1] implements the procedure $\texttt{Find}_y$ which finds the index of a point in $\texttt{array}_y$: if such a procedure is implemented using the hash table, as claimed, then each removal of a point from $\texttt{array}_y$ shall result in an update of $\Theta(n)$ entries of that hash table on average. For our analysis we optimistically assume that both $\texttt{Find}_y$ and removal of a point from an array take $O(1)$ time.

Second, an important observation about LONSA is that, according to [1], the points are sorted by the first and the second objectives, to obtain $\texttt{array}_x$ and $\texttt{array}_y$ correspondingly, while essentially ignoring third and all higher objectives. For this reason, we consider the case where the values of the first objective are the same across all the points, and the same property holds for the second objective. In this case, LONSA does not reorder the supplied points, so the values of all the objectives can be arbitrary.

Under the assumption above, the indices of every point in arrays $\texttt{array}_x$ and $\texttt{array}_y$ coincide, and LONSA behaves in objectives $[3..M]$ exactly like another non-dominated sorting algorithm, Deductive Sort [3]. For this algorithm, the inputs of size N exist, as shown in a recent paper [4], for any number of objectives $M' \geq 2$ which makes it perform $\Theta(N^3)$ dominance comparisons, where each of the comparisons can be further forced to require $\Theta(M)$ time to determine the dominance relation. Hence Deductive Sort has the worst-case running time of $\Theta(MN^3)$. For the reasons explained above, LONSA will also require at least the same time if one prepends two equal objective values to each point from such an input. Hence, the worst-case time complexity of LONSA is also $\Theta(MN^3)$. This disproves the claim of the authors of LONSA about having $O(N^2)$ dominance comparisons in the worst case, which they made in [1, Section 3.3].

3 CONCLUSION

We presented a short proof that LONSA, a recently proposed algorithm for non-dominated sorting, has the worst-case running time of $\Theta(MN^3)$. Our proof used a deficiency in this algorithm: whenever the first two objectives of all points are the same, it degenerates into another algorithm, Deductive Sort, which is, in turn, possible to force into $\Theta(N^3)$ dominance comparisons.

With this small paper, we once more remind that those claims about the performance of the algorithms, which can be easily validated or disproved, shall be proven, not just stated as the authors of LONSA did in their complexity analysis.

ACKNOWLEDGMENTS

This work is partially financially supported by National Center for Cognitive Research of ITMO University.

REFERENCES

[1] Rafael Frederico Alexandre, Carlos Henrique Nogueira de Resende Barbosa, and João Antônio de Vasconcelos. 2018. LONSA: A Labeling-Oriented Non-Dominated Sorting Algorithm for Evolutionary Many-Objective Optimization. *Swarm and Evolutionary Computation* 38 (2018), 275–286.

[2] 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.

[3] 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.

[4] 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.

[5] N. Srinivas and Kalyanmoy Deb. 1994. Multiobjective Optimization Using Non-dominated Sorting in Genetic Algorithms. *Evolutionary Computation* 2, 3 (Fall 1994), 221–248.

[6] Sorrachai Yingchareonthawornchai, Proteek Chandan Roy, Bundit Laekhanukit, Eric Torng, and Kalyanmoy Deb. 2020. Worst-case conditional hardness and fast algorithms with random inputs for non-dominated sorting. In *Proceedings of Genetic and Evolutionary Computation Conference Companion*. ACM, 185–186.