An Efficient Nondominated Sorting Algorithm

Junchen Wang
School of Automation, China University of Geosciences
Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems
Wuhan, China
wangjunchen@cug.edu.cn

Changhe Li*
School of Automation, China University of Geosciences
Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems
Wuhan, China
changhe.lw@gmail.com

Yiya Diao
School of Automation, China University of Geosciences
Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems
Wuhan, China

Sanyou Zeng
School of Mechanical Engineering and Electronic Information, China University of Geosciences
Wuhan, China

Hui Wang
Nanchang Institute of Technology
Nanchang, China

ABSTRACT

Nondominated sorting (NS) is commonly needed in multi-objective optimization to distinguish the fitness of solutions. Since it was suggested, several NS algorithms have been proposed to reduce its time complexity. In our study, we found that their performances are closely related to properties of the distribution of a data set, especially the number of fronts. To address this issue, we propose a novel NS algorithm *Filter Sort*. We also propose a new benchmark data generator for evaluating the performance of a NS algorithm. Experimental results show that our algorithm is superior to several state-of-the-art NS algorithms in most cases.

CCS CONCEPTS

• **Theory of computation** → **Sorting and searching**; *Database query processing and optimization (theory)*; • **Mathematics of computing** → Mathematical optimization;

KEYWORDS

Multi-objective optimization, nondominated sorting, filter sort

ACM Reference Format:
Junchen Wang, Changhe Li, Yiya Diao, Sanyou Zeng, and Hui Wang. 2018. An Efficient Nondominated Sorting Algorithm. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205663

1 INTRODUCTION

Since Srinivas and Deb proposed a NS algorithm in 1995, several NS algorithms have been proposed to reduce the time complexity even further. In this paper, we propose a novel NS algorithm namely *Filter Sort*, which could save a lot of comparisons when the number of fronts is either small or big. We also propose a benchmark generator, which is able to generate test data sets with a user-defined number of Pareto fronts, to help prove our algorithm's superiority.

2 *FILTER SORT*

In *Filter Sort*, we firstly sort solutions by each objective value. Solution $\boldsymbol{x_i}$'s sequence number on the $j\ th$ objective is denoted by $s_{i,j}$. We sum up all its objective sequence numbers and term the sum as S_i. We define a filter solution as the solution which has the minimum value of S. It is easy to prove that the filter solution is not dominated by any other solution in a data set.

Then we find out all solutions that have at least one objective's value which is smaller than that of the filter solution, and termed them as candidates. Although all candidate solutions are not dominated by the filter solution, there may exist candidate solutions that are dominated by other candidate solutions.

When we check whether a candidate is dominated by others, we only compare it with solutions whose objective value is smaller than that of it on its best objective. In addition, in this process , as we only want to know whether the candidate is dominated or not, we first choose the comparing solution's worst objective to compare, thus speeding up this comparison,

After we find solutions of the first front, we remove them from a data set, and repeat these processes until no solution exists. The pseudo code of *Filter Sort* is shown in Algorithm 1, where the index of the best objective of solution $\boldsymbol{x_i}$ is denoted by B_i, the index of the worst objective of solution $\boldsymbol{x_i}$ is denoted by W_i, and $\boldsymbol{filters}[i]$ denotes the index of solution which has the i th smallest S, and also ***Cand*** stores candidates, while ***Comp*** stores comparisons.

3 EXPERIMENTAL RESULTS

In the following experiments, we choose four state-of-the-art NS algorithms *Deductive Sort* [1], *Corner Sort* [2] and *T-ENS*[3] as peer algorithms, all of which were programmed in C++. All experiments in this paper were conducted on a PC with 3.6GHz intel Core i7-7700 CPU and memory of 16G in Windows 10 (64bit).

*The corresponding author

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3205663

ALGORITHM 1: *Filter Sort*

Input: Data set $\boldsymbol{P}$, number of solutions N, number of objectives M
Output: Front sets $\boldsymbol{R}$
foreach $j \in [1:M]$ **do**
  $\{s_{i,j} \mid i \in [1:N]\} \leftarrow quick\_sort(\{x_{i,j} \mid i \in [1:N]\})$;
  store all solutions' indexes in sequence within $\boldsymbol{LinkedList_j}$;
foreach $i \in [1:N]$ **do**
  calculate S_i; find B_i, W_i;
$\boldsymbol{filters} \leftarrow quick\_sort(\{S_i \mid i \in [1:N]\})$;
$FrontNum$:=1; $NumRanked$:=0;
while $NumRanked < N$ **do**
  $filter$:=$\boldsymbol{filters}$[1];
  foreach $j \in [1:M]$ **do**
    $\boldsymbol{Cand} \leftarrow$ traverse the $\boldsymbol{LinkedList_j}$ from begin to $filter$;
  $\boldsymbol{R}[filter]$:=$FrontNum$; $NumRanked$++;
  delete $filter$ from $\boldsymbol{filters}$ and all $\boldsymbol{LinkedLists}$;
  foreach $c \in \boldsymbol{Cand}$ **do**
    $dominated$:=$true$;
    $\boldsymbol{Comp} \leftarrow$ traverse the $\boldsymbol{LinkedList_{B_c}}$ from begin to c;
    foreach $d \in \boldsymbol{Comp}$ **do**
      if $s_{(d,W_d)} > s_{(c,W_d)}$ **then**
        continue;
      else if $\forall j \in [1, M](j \neq W_d), s_{d,j} < s_{c,j}$ **then**
        $dominated$:=$false$
    if $dominated = false$ **then**
      $\boldsymbol{Cand}$−=c;
  foreach $c \in \boldsymbol{Cand}$ **do**
    $\boldsymbol{R}[c]$:=$FrontNum$; $NumRanked$++;
    delete c from $\boldsymbol{filters}$ and all $\boldsymbol{LinkedLists}$;
  $FrontNum$++;
return $\boldsymbol{R}$;

In the literature, the number of fronts of benchmark data can not be controlled, so we propose a new benchmark generator, which can generate a set of solutions with not only a user-defined number of solutions, and a user-defined number of objectives, but also a user-defined number of fronts.

Figure 1 presents the changes of the runtime of four algorithms as the increase of the number of fronts on a data set with different combinations of the number of solutions and the number of objectives. Each algorithm remains almost the same tendency when the number of solutions and the number of objectives are different. The runtime of *Deductive Sort* and *Corner Sort* decrease with the increase of the number of fronts, while the runtime of *T-ENS* increases with the increase of the number of fronts.

We can also see that, *Filter Sort* outperforms the other three algorithms in most cases. A plausible explanation is that, a filter solution can find out fewer candidates than a corner solution. Another reason is that, two tricks in the process of checking candidates can effectively save comparisons when the number of objectives is big.

4 CONCLUSIONS

In this paper, we propose a novel NS algorithm namely *Filter Sort* . We also propose a new benchmark generator to generate data sets with a user-defined number of fronts. Experimental results show that *Filter Sort* is superior to several state-of-the-art NS algorithms in most cases.

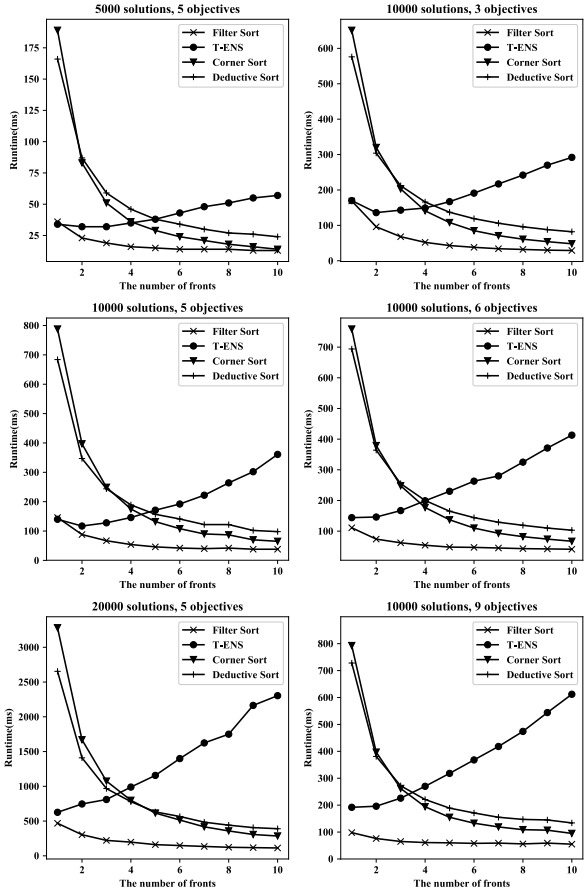

Figure 1: Runtime of four NS algorithms on randomly generated data sets with different combinations of the number of solutions and the number of objectives.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant 61673355, the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, the 111 project under Grant B17040, and the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) under Grant 201705.

REFERENCES

[1] K. McClymont and E. Keedwell. 2012. Deductive Sort and Climbing Sort: New Methods for Non-Dominated Sorting. *Evolutionary Computation* 20, 1 (2012), 1–26.
[2] H. Wang and X. Yao. 2014. Corner Sort for Pareto-Based Many-Objective Optimization. *IEEE Transactions on Cybernetics* 44, 1 (2014), 92–102.
[3] X. Zhang, Y. Tian, R. Cheng, and Y. Jin. 2018. A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale Many-objective Optimization. *IEEE Transactions on Evolutionary Computation* 22, 1 (2018), 97–112.