Improving preference learning for MR-Sort using GPU

Laurent Cabaret, Vincent Jacques, Vincent Mousseau[1]

Abstract. The Majority Rule Sorting (MR-Sort) method assigns alternatives evaluated on multiple criteria to one of the predefined ordered categories. The Inverse MR-Sort problem (Inv-MR-Sort) consists in computing MR-Sort parameters that match a dataset. Although Inv-MR-Sort is known to be computationally difficult, exact resolution approaches have been proposed in the literature, but are confronted to a computational barrier. In contrast, Sobrie et al. [12] tackled it with a heuristic. In this work, we aim at improving the computational efficiency of this heuristic approach by parallelization strategies using GPU.

1 Introduction

In this paper, we consider multiple criteria sorting problems in which alternatives evaluated on several criteria are to be assigned to one of the pre-defined ordered categories C^1, C^2, ..., C^p, C^1 (C^p, respectively) being the worst (best, respectively) category.

Many multiple criteria methods have been proposed in the literature (see e.g. [5],[15]). We are interested in a pairwise comparison based method: the Non-Compensatory Sorting model (NCS, see [2, 3]). NCS assigns alternatives to categories based on the way alternatives compare to boundary profiles representing frontiers between consecutive categories and can be viewed as an axiomatic formulation of the Electre Tri method (see [11]). More specifically, we consider a particular case of NCS in which the importance of criteria is additively represented using weights: the Majority Rule Sorting (MR-Sort, see [7]).

In real-world decision problems involving multiple criteria sorting, the implementation of a sorting model requires eliciting the decision-maker's (DM) preferences and adequately representing her preferences by setting appropriate values for the preference-related parameters. It is usual to elicit the sorting model parameters indirectly from a set of assignment examples, i.e., a set of alternatives with corresponding desired categories. Such preference learning approach has been developed for MR-Sort (Inv-MR-Sort, see, e.g. [7], [13]), and makes it possible to compute MR-Sort parameters that best fit a learning set provided by the DM.

2 NCS, MR-Sort model and Inv-MRSort

2.1 NCS and MR-Sort

Non-compensatory Sorting (NCS) [2, 3] is an MCDA sorting model originating from the ELECTRE TRI method [6]. NCS can be intuitively formulated as follows: an alternative is assigned to a category if *(i)* it is better than the lower limit of the category on a sufficiently strong subset of criteria, and *(ii)* this is not the case when comparing the alternative to the upper limit of the category.

Consider the simplest case involving 2 categories Good ($\mathcal{G}$) and Bad ($\mathcal{B}$) with the following notations. We denote X_i the finite set of possible values on criterion i, $i \in \mathcal{N} = \{1, \ldots, n\}$; we suppose w.l.o.g. that $X_i = [min_i, max_i] \subset \mathbb{R}$. Hence, $X = \prod_{i \in \mathcal{N}} X_i$ represents the set of alternatives to be sorted. We denote $\mathcal{A}_i \subseteq X_i$ the set of approved values on criterion $i \in \mathcal{N}$. Approved values on criterion i ($x_i \in \mathcal{A}_i$) correspond to values contributing to the assignment of an alternative to category $\mathcal{G}$. In order to assign alternative a to category $\mathcal{G}$, a should have approved values on a subset of criteria which is "sufficiently strong". The set $\mathcal{F} \subseteq 2^{\mathcal{N}}$ contains the "sufficiently strong" subsets of criteria; it is a subset of $2^{\mathcal{N}}$ up-closed by inclusion. In this perspective, the NCS assignment rule can be expressed as follows:

$$x \in \mathcal{G} \quad \text{iff} \{i \in \mathcal{N} : x_i \in \mathcal{A}_i\} \in \mathcal{F}, \quad \forall x \in X \tag{1}$$

With more than two categories, we consider an ordered set of p categories $C^p \rhd \cdots \rhd C^h \rhd \cdots \rhd C^1$, where $\rhd$ denotes the order on categories. Sets of approved values $\mathcal{A}_i^h \subseteq X_i$ on criterion i ($i \in \mathcal{N}$) are defined with respect to a category h ($h = 2..p$), and should be defined as embedded sets such that $\mathcal{A}_i^2 \supseteq ... \supseteq \mathcal{A}_i^p$. Analogously, sets of sufficiently strong criteria coalitions are relative to a category h, and are embedded as follows: $\mathcal{F}^2 \supseteq ... \supseteq \mathcal{F}^p$. The assignment rule is defined bellow, for all $x \in X$, where $\mathcal{A}_i^1 = X_i$, $\mathcal{A}_i^{p+1} = \emptyset$, $\mathcal{F}^1 = \mathcal{P}(\mathcal{N})$, and $\mathcal{F}^{p+1} = \emptyset$.

$$x \in C^h \quad \text{iff} \quad \begin{array}{l} \{i \in \mathcal{N} : x_i \in \mathcal{A}_i^h\} \in \mathcal{F}^h \text{ and} \\ \{i \in \mathcal{N} : x_i \in \mathcal{A}_i^{h+1}\} \notin \mathcal{F}^{h+1} \end{array} \tag{2}$$

A particular case of NCS corresponds to the MR-Sort rule [7]. When the families of sufficient coalitions are all equal $\mathcal{F}^2 = ... = \mathcal{F}^p = \mathcal{F}$ and defined using additive weights attached to criteria, and a threshold: $\mathcal{F} = \{F \subseteq \mathcal{N} : \sum_{i \in F} w_i \geq \lambda\}$, with $w_i \geq 0$, $\sum_i w_i = 1$, and $\lambda \in [0, 1]$. Moreover, as the finite set of possible values on criterion i, $X_i = [min_i, max_i] \subset \mathbb{R}$, the order on $\mathbb{R}$ induces a complete pre-order $\succcurlyeq_i$ on X_i. Hence, the sets of approved values on criterion i, $\mathcal{A}_i^h \subseteq X_i$ ($i \in \mathcal{N}, h = 2...p$) are defined by $\succcurlyeq_i$ and $b_i^h \in X_i$ the minimal approved value in X_i at level h: $\mathcal{A}_i^h = \{x_i \in X_i : x_i \succcurlyeq_i b_i^h\}$. In this way, $b^h = (b_1^h, \ldots, b_n^h)$ is interpreted as the frontier between categories C^{h-1} and C^h; $b^1 = (min_1, ..., min_n)$ and $b^{p+1} = (max_1, ..., max_n)$ are the lower frontier of C^1 and the upper frontier of C^p. Therefore, the MR-Sort rule can be expressed as:

$$x \in C^h \quad \text{iff} \quad \sum_{i : x_i \geq b_i^h} w_i \geq \lambda \text{ and } \sum_{i : x_i \geq b_i^{h+1}} w_i < \lambda \tag{3}$$

2.2 Inv-MR-Sort

MR-Sort preference parameters, e.g. weights, majority level, and limit profiles, can be either initialized by the "end-user", i.e. the

[1] MICS, CentraleSupélec, Université Paris-Saclay
laurent.cabaret@centralesupelec.fr
vincent@vincent-jacques.net
vincent.mousseau@centralesupelec.fr

decision-maker, or learned through a set of assignment examples called a learning set. We are focusing on the learning approach. The aim is to find the MR-Sort parameters that "best" fit the learning set.

We consider as input a learning set, denoted L, composed of assignment examples. Here, an *assignment example* refers to an alternative $a \in A^\star \subset X$, and a desired category $c(a) \in \{1, \ldots, p\}$. In our context, the determination of MR-sort parameters values relies on the resolution of a mathematical program based on assignment examples: the *Inv-MR-Sort* problem takes as input a learning set L and computes weights $(w_i, i \in \mathcal{N})$, majority level (λ), and limit profiles $(b_h, h = 2..p)$ that best restore L, i.e. that maximizes the number of correct assignments.

This learning approach – also referred to as preference disaggregation – has been previously considered in the literature. In particular, [10],[14] learned the ELECTRE TRI parameters using mathematical programming formulation (non-linear programming for the former, mixed-integer programming for the latter). In contrast, [4] propose an evolutionary approach to do so.

Later, a more amenable model, the MR-Sort – which derives from the ELECTRE TRI method and requires fewer parameters than ELECTRE TRI – was introduced by Leroy et al. in [7]. They proposed a MIP implementation for solving the *Inv-MR-Sort* problem. Belahcene et al. [1] tackled it with a Boolean satisfiability (SAT) formulation. In contrast with these exact formulations, Sobrie et al. [12] tackled it with a heuristic; indeed, as inv-NCS, Inv-MR-Sort is known to be computationally difficult.

3 Improving Sobrie's heuristic on GPUs

In this paper, we aim at improving the efficiency of Sobrie's heuristic [12] through the use of new parallel architectures. The current state of the art make it possible to handle (with a computation time compatible with an interaction with the DM) datasets involving up to 10 criteria, 5 categories, and several hundreds of assignment examples. We aim a being able to consider up to several dozen of criteria, and several thousands of examples (with 5 categories), which would open the path to new types of application.

Hence, in our approach, the main objective is to benefit from the performance of computer with parallel architecture in order to improve the computational performance of the Inv-MR-Sort resolution. To migrate the In-MR-Sort heuristic proposed by Sobrie et al [13] that was originally implemented in Python, we started by re-implementing the heuristic in C. Then we used the CUDA general purpose parallel computing platform to propose an optimized version.

4 Experimental results

We performed tests which aims at evaluating the comparative performance of Inv-MR-Sort implemented in Python on CPU, C on CPU, and C on GPU. Tests were performed on a machine (here is the description...).

In the numerical investigations, we vary the number of criteria $(n \in [6, 20])$, the number of categories $(p = 2, 3, 4, 5)$, and set the size of the learning set to 10000. We repeat each instance size 10000 times. To generate a ground truth, we randomly generate an MR-Sort model $\mathcal{M}^0$, and randomly generate n-tuples of values considered as alternatives. Then we simulate the assignments of these alternatives using $\mathcal{M}^0$. The obtained learning set L, used as input to the three implementations of Inv-MR-Sort (Python-CPU, C-CPU, and C-GPU), generating a learned model noted $\mathcal{M}'$. For each instance, we observe the computing time, and compute the generalisation on a randomly generated test set of 10^6 alternatives.

The result show the benefit of solving Inv-MR-Sort problem instances on a parallel architecture and makes it possible to consider the resolution of larger instances size in a timing compatible with an interaction with a decision-maker.

5 Conclusion and further research

In this paper, we developed an GPU implementation of the heuristic for Inv-MR-Sort proposed by Sobrie et al. [12]. Our work show that the use the capabilities of the new parallel architecture enables to consider datasets of larger size, which opens new application area.

To further extend the type of applications, research should be continued in order to consider single peaked preference on criteria (see [8, 9]). Another research avenue that arize when the number of criteria increases, is related to the computation of a "parcimonious" MR-Sort representation of the learning set, involving only a limited subset of criteria.

REFERENCES

[1] K. Belahcene, C. Labreuche, N. Maudet, V. Mousseau, and W. Ouerdane, 'An efficient SAT formulation for learning multiple criteria non-compensatory sorting rules from examples', *Computers & Operations Research*, **97**, 58–71, (2018).

[2] D. Bouyssou and T. Marchant, 'An axiomatic approach to noncompensatory sorting methods in mcdm, i: The case of two categories', *European Journal of Operational Research*, **178**, 217–245, (2007).

[3] D. Bouyssou and T. Marchant, 'An axiomatic approach to noncompensatory sorting methods in MCDM, II: More than two categories', *European Journal of Operational Research*, **178**(1), 246–276, (2007).

[4] M. Doumpos, Y. Marinakis, M. Marinaki, and C. Zopounidis, 'An evolutionary approach to construction of outranking models for multicriteria classification: The case of the ELECTRE TRI method', *European Journal of Operational Research*, **199**(2), 496–505, (2009).

[5] M. Doumpos and C. Zopounidis, *Multicriteria Decision Aid Classification Methods*, Kluwer Academic Publishers, Dordrecht, 2002.

[6] J. Figueira, V. Mousseau, and B. Roy, 'ELECTRE methods', in *Multiple criteria decision analysis: State of the art surveys*, 133–153, Springer, (2005).

[7] A. Leroy, V. Mousseau, and M. Pirlot, 'Learning the parameters of a multiple criteria sorting method', in *International Conference on Algorithmic Decision Theory*, pp. 219–233. Springer, (2011).

[8] P. Minoungou, *Learning Majority-Rule models with partially monotone data*, Ph.D. dissertation, Université Paris-Saclay, 2022.

[9] P. Minoungou, V. Mousseau, W. Ouerdane, and P. Scotton, 'Learning MR-Sort models from non-monotone data', *Annals of of Operational Research*, (to appear).

[10] V. Mousseau and R. Slowinski, 'Inferring an ELECTRE TRI model from assignment examples', *Journal of global optimization*, **12**(2), 157–174, (1998). Springer.

[11] B. Roy, 'The outranking approach and the foundations of Electre methods', *Theory and Decision*, **31**(1), 49–73, (1991).

[12] O. Sobrie, *Learning preferences with multiple-criteria models*, Ph.D. dissertation, Université de Mons (Faculté Polytechnique) and Université Paris-Saclay (CentraleSupélec), 2016.

[13] O. Sobrie, V. Mousseau, and M. Pirlot, 'Learning monotone preferences using a majority rule sorting model', *Int. Trans. Oper. Res.*, **26**(5), 1786–1809, (2019).

[14] J. Zheng, S. Metchebon Takougang, V. Mousseau, and M. Pirlot, 'Learning criteria weights of an optimistic Electre Tri sorting rule', *Computers and Operations Research*, **49**, 28 – 40, (2014).

[15] C. Zopounidis and M. Doumpos, 'Multicriteria classification and sorting methods: A literature review', *European Journal of Operational Research*, **138**(2), 229–246, (2002).