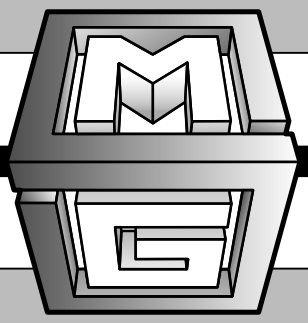

Université Libre de Bruxelles

CoDE - SMG

FlowSort parameters elicitation: the case of partial sorting

CoDE-SMG – Technical Report Series

Dimitri Van Assche, Yves De Smet

CoDE-SMG – Technical Report Series

Technical Report No.
TR/SMG/2014-006

September 2014

CoDE-SMG – Technical Report Series
ISSN 2030-6296

Published by:
CoDE-SMG, CP 210/01
Université Libre de Bruxelles
Bvd du Triomphe
1050 Ixelles, Belgium

Technical report number TR/SMG/2014-006

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of CoDE-SMG. The authors take full responsability for any copyright breaches that may result from publication of this paper in the CoDE-SMG – Technical Report Series. CoDE-SMG is not responsible for any use that might be made of data appearing in this publication.

FlowSort parameters elicitation: the case of partial sorting

CoDE-SMG – Technical Report Series

Dimitri Van Assche dimitri.van.assche@ulb.ac.be
Yves De Smet yves.de.smet@ulb.ac.be
CoDE-SMG, Université Libre de Bruxelles, Brussels, Belgium

September 2014

FlowSort parameters elicitation: the case of partial sorting

Dimitri Van Assche[1], **Yves De Smet**[2]

Abstract. We consider the context of partial sorting. We address the problem of finding the parameters of the FlowSort method using an existing categorization. This contribution constitutes an extension of a method we have developed in the context of complete sorting. It relies on the use of a dedicated Genetic Algorithm based on variations of search parameters. We show how to manage the problem of correct categorization prediction, which is more difficult, since ranges of categories are considered. The method is tested on three different datasets for which a partial sorting has been generated with a particular instantiation of FlowSort.

1 Introduction

Multi-criteria decision aid (MCDA) has been an active research field for more than 40 years. In this context, possible decisions are simultaneously evaluated on multiple conflicting criteria. For instance, in the common example of buying a new car, one typically tries to minimize the cost and consumption while maximizing performances, comfort, etc. Obviously no real car would be the best on all those criteria. Therefore, the notion of optimal solution is most of time replaced by the idea of compromise solution [8].

In this paper, we will work with the well-known outranking method PROMETHEE [1]. We focus on the sorting problem, i.e. the assignment of alternatives into predefined categories. For instance, sorting countries into risk categories on the basis of economical, financial and political indicators. In this paper, we work with FlowSort, which is a natural extension of PROMETHEE for sorting problems. [7]

In the context of FlowSort, the decision maker needs to give central, or limit, profiles defining each category and preference parameters characterizing each criterion. Here, we consider the problem inside out: based on an existing categorization, one tries to find the parameters of FlowSort, which allow to best replicate the existing categorization.

Let us point out that we have recently submitted a first contribution on the preference elicitation for FlowSort based on assignment examples [10]. This paper only considers complete sorting problems: each alternative is assumed to belong to a unique category. In between, we have slightly improved the performances of the Genetic Algorithm we used.

Furthermore, we propose an extension of this first work to deal with partial sorting. The idea is that an alternative may belong to different categories at the same time. Here after, we propose an adapted method to elicit the preferences in the case of partial sorting.

In section 2, we introduce PROMETHEE and FlowSort. In section 3, we describe the genetic algorithm we use to solve the related optimization problem. Then, in section 4, we illustrate the algorithm on different datasets for which a categorization has been computed with a particular instantiation of FlowSort. Finally, we present the performances of our algorithm in the case of partial sorting.

2 PROMETHEE and FlowSort

In this section, we briefly present PROMETHEE[3] I and II as well as FlowSort. For additionnal information, we refer the interested reader to [3] for a detailed description of PROMETHEE and to [6] for FlowSort.

Let $A = \{a_1, a_2..., a_n\}$ be a set of n alternatives and let $F = \{f_1, f_2..., f_q\}$ be a family of q criteria. The evaluation of alternative a_i for criterion l will be denoted by a real value $f_l(a_i)$.

For each pair of alternatives, let's compute $d_l(a_i, a_j)$, the difference of a_i over a_j on criterion l.

$$d_l(a_i, a_j) = f_l(a_i) - f_l(a_j) \tag{1}$$

A preference function, denoted P_l, is associated to each criterion l. This function transforms the difference of alternatives' evaluations $d_l(a_i, a_j)$ into a preference degree of the first alternative over the second one for criterion l. Without loss of generality, we consider that criteria have to be maximized. P_l is defined as follow:

$$P_l : \mathbb{R} \to [0, 1] : x \to P_l(x) \tag{2}$$

such that:

- $\forall x \in \mathbb{R}^- : P_l(x) = 0$,
- $\forall x, y \in \mathbb{R}_0^+ : x \leq y \implies P_l(x) \leq P_l(y)$

There are different kinds of preference functions. Henceforth, we consider only the linear one (see figure 1) which is characterized by two parameters: an indifference and a preference threshold: q_l, p_l.

$$\pi_l(a_i, a_j) = P_l[d_l(a_i, a_j)] = \begin{cases} 0 & \text{if } d_l(a_i, a_j) \leq q_l \\ \frac{d_l(a_i, a_j) - q_l}{p_l - q_l} & \text{if } q_l < d_l(a_i, a_j) \leq p_l \\ 1 & \text{if } p_l < d_l(a_i, a_j) \end{cases} \tag{3}$$

Once $\pi_l(a_i, a_j)$ has been computed for all pairs of alternatives, we may define the aggregated preference degree of alternative a_i over a_j using the weights w_l associated to each criterion l. Weights are assumed to be positive and normalized.

[1] Computer & Decision Engineering (CoDE), Université libre de Bruxelles, email: dvassche@ulb.ac.be

[2] Computer & Decision Engineering (CoDE), Université libre de Bruxelles, email: yvdesmet@ulb.ac.be

[3] Preference Ranking Organization METHod for Enrichment of Evaluations

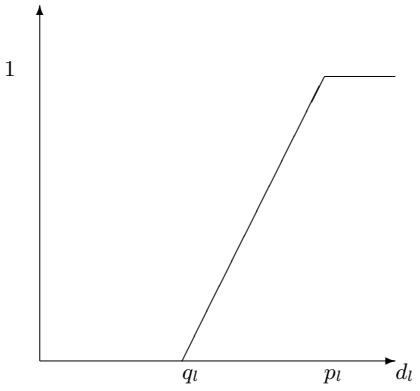

Figure 1. Linear preference function

$$\pi(a_i, a_j) = \sum_{l=1}^{q} w_l.\pi_l(a_i, a_j) \tag{4}$$

The last step consists in calculating the positive flow score denoted $\phi_A^+(a_i)$ and the negative flow score denoted $\phi_A^-(a_i)$ as follow:

$$\phi_A^+(a_i) = \frac{1}{n-1} \sum_{x \in A} \pi(a_i, x) \tag{5}$$

$$\phi_A^-(a_i) = \frac{1}{n-1} \sum_{x \in A} \pi(x, a_i) \tag{6}$$

We define the net flow score of a_i as the difference between the positive flow and negative flows of a_i:

$$\phi_A(a_i) = \phi_A^+(a_i) - \phi_A^-(a_i) \tag{7}$$

The PROMETHEE I ranking is obtained as the intersection of the rankings induced by ϕ^+ and ϕ^-. For an interpretation of the net flow scores, the interested reader is referred to [5]. Finally, a complete order, called PROMETHEE II, can be derived from the order induced by ϕ.

Based on PROMETHEE, FlowSort has been developed to address sorting problems [6]. Let $C = \{c_1, c_2..., c_k\}$ be a set of k ordered categories. We assume that $c_i \succ c_{i+1}$: c_i is preferred to c_{i+1}. Therefore C_1 is the best category and C_k is the worst one.

Categories are assumed to be represented by limit or central profiles. On the one hand, the idea of the limiting profiles is to define couples of values for each criterion, defining the lower and upper bounds of the considered category. Let us note that the profile defining the upper bound of category c_i is the same as the one defining the lower bound of category c_{i+1}. On the other hand, central profiles are defined using a single value for each criterion. This represents a kind of mean profile of the category. A common property is that the profiles of each category must dominate the profiles of the ones they are preferred to. In this work, we have chosen to work with central profiles.

Let's define $R = \{r_1, r_2..., r_k\}$, the set of central profiles representing the k categories. To identify the category of an alternative a_i, we define the subset $R_i = R \cup \{a_i\}$. Then, for each element x in the subset R_i, we compute its net flow score $\phi_{R_i}(x)$.

As in the nearest neighbor procedure, the category of alternative a_i is the one such that the profile has its net flow score the closest to the net flow score of a_i. More formally:

$$l^*(a_i) = \underset{l=1,2...,k}{argmin} |\phi_{R_i}(a_i) - \phi_{R_i}(r_l)| \tag{8}$$

Let us note that $(3+k).q$ parameters have to be provided in order to instantiate FlowSort:

- $k.q$ values for the central profiles;
- $3.q$ values for the weights, indifference and preference thresholds.

In the case of the partial sorting problem, an alternative can be sorted in multiple consecutive categories. In this case, we use an extension of PROMETHEE I instead of PROMETHEE II. The upper and lower categories are determined using the positive and negative flow scores. As in the regular FlowSort method, the category of a_i is determined as the category of the profile having its positive, resp. negative, flow score the closest to the the one of the alternative a_i.[6]

$$l_+^*(a_i) = \underset{l=1,2...,k}{argmin} |\phi_{R_i}^+(a_i) - \phi_{R_i}^+(r_l)| \tag{9}$$

$$l_-^*(a_i) = \underset{l=1,2...,k}{argmin} |\phi_{R_i}^-(a_i) - \phi_{R_i}^-(r_l)| \tag{10}$$

If both values are equal, the categorization is precise. Otherwise, these values define the range of the categories for the given alternative.

3 Algorithms

The algorithms developed to learn the FlowSort parameters are the same as those presented in [10]. This approach is based on a dedicated genetic algorithm. Henceforth we only present the specific points that are dedicated to partial sorting i.e.:

- the definition of a distance measure in order to guide the optimization process;
- the evaluation of the correctness of a particular solution.

Compared to the previous approach [10], we have also completely changed the parameters optimization process. In [10], we have used iRace[4] in order to fine tune the parameters of the algorithm. For more information on the iRace procedure, we refer the interested reader to [4] [2]. We will describe the new procedure hereafter.

A distinctive feature of partial sorting is that we have to deal with two pieces of information: the upper and lower categories for each alternative. Let us note $c^+(a_i)$ the upper category and $c^-(a_i)$ the lower one of an alternative a_i. We have chosen to use the L_1 distance between the upper and lower category given as input c_r and the one given by the current parametrization of FlowSort c_f. Hence, s will denote the current parameters vector (a current solution). The distance is used to induce a higher penalty if there is a big difference between the prediction and the real category. More formally, the penalty associated to s is defined as follow:

$$f(s) = \sum_{a \in A} (|c_f^+(a) - c_r^+(a)| + |c_f^-(a) - c_r^-(a)|) \tag{11}$$

Let us recall that this distance is used to guide the optimization process [10].

The correctness defines how good a solution is with respect to the real categorization. The correctness of a single alternative is defined as the number of categories correctly predicted divided by the total

[4] Iterated Race for Automatic Algorithm Configuration

range covered by the predicted and the real categories. The correctness of a solution is defined as the sum of the correctness of all the alternatives:

$$corr(s) = \sum_{a \in A} \frac{min(c_f^+(a), c_r^+(a)) - max(c_f^-(a), c_r^-(a)) + 1}{max(c_f^+(a), c_r^+(a)) - min(c_f^-(a), c_r^-(a)) + 1} \quad (12)$$

The identification of the best possible solution regarding the correctness is based on a genetic algorithm. This algorithm has mainly two kind of exploration: diversification with the mutation operator and intensification with the crossover operator. During the tests, we have observed that the algorithm should ideally enforce diversification after intensification and then go back to intensification, and so on. As a consequence, the idea we have applied is to force the parameters variation of the algorithm during the optimization process. There are 5 parameters: population size, mutation probability, gene mutation probability, crossover probability, gene crossover probability. The population size has been fixed to 1600 solutions. This value has been fixed after a set of trial and errors, and seems to work well in the considered examples. The 4 others parameters have values between 0 and 1. At each step of the optimization we change the values of those following a linear equation. When the value 0, or 1, is reached the coefficient is reversed. We paid attention to chose different coefficients, so that the period is different. This permits to have a lot of different combinations of intensification and combinations. As illustrated on:

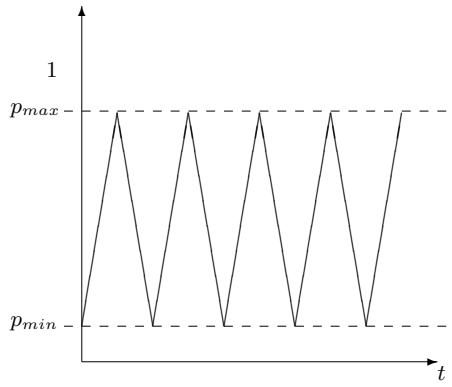

Figure 2. Varying parameters for the GA

In table 1, we show the values we have chosen for p_{min} and p_{max} for each paramter.

parameter	p_{min}	p_{min}
mutation probability	0.1	0.9
gene mutation probability	0.25	0.99
crossover probability	0.1	0.9
gene crossover probability	0.25	0.99

Table 1. Values of p_{min} and p_{max} for each parameter.

With this new method, we have seen a slight improvement of the results compared to our previous work. We were able to increase the correctness of the prediction applied to the learning set. Unfortunately, this did not really improve the prediction rate using a testing set.

4 Results

In [10], we have worked on 3 real datasets for the validation: CPU, BC and CEV. They come from the website of Marburg University[5]. These originally come from the UCI repository[6] and the WEKA machine learning toolbox[7]. Let us point out that these datasets have also been used by Sobrie [9] in the context of complete sorting but using a modified version of ELECTRE TRI.

To the best of our knowledge, there is no dataset for partial sorting. As a consequence, we decided to use the same 3 datasets but generating a partial categorization by using a random instantiation of FlowSort. Therefore we know an exact solution exists for the model's parameters. The properties of the datasets are in table 2.

dataset	#inst.	#crit.	#cat.	% imprecise cat.
CPU	209	6	4	40.67
BC	278	7	2	23.02
CEV	1728	6	4	16.43

Table 2. Datasets used for the tests

The testing procedure has been set as follow: each dataset has been divided in a learning set and a test set. Different sizes of learning set have been considered. Alternatives in the learning set have been randomly selected. Nevertheless, we forced the algorithm to select randomly at least one alternative from each category. For each learning and test set, the algorithm has been executed on the learning set to elicit the parameters. Then they found have been evaluated on the test set. This operation has been executed 32 times for each learning set. For robustness' sake, the whole operation has been executed 10 times for each value of the learning set size. The maximum number of evaluations has been set to 2 500 000, and the population size to 1600. Results are avaiable in table 3.

learning set's size	dataset	correctness	learning set correctness
5%	CPU	0.7337 ± 0.0705	1.0000 ± 0.0000
	BC	0.8827 ± 0.0337	0.9981 ± 0.0120
	CEV	0.8498 ± 0.0224	0.9227 ± 0.0383
20%	CPU	0.8798 ± 0.0245	0.9880 ± 0.0160
	BC	0.9463 ± 0.0209	0.9955 ± 0.0103
	CEV	0.8809 ± 0.0173	0.8554 ± 0.0338
35%	CPU	0.9004 ± 0.0215	0.9642 ± 0.0243
	BC	0.9579 ± 0.0210	0.9919 ± 0.0110
	CEV	0.8868 ± 0.0154	0.8395 ± 0.0277
50%	CPU	0.9065 ± 0.0228	0.9581 ± 0.0214
	BC	0.9747 ± 0.0163	0.9913 ± 0.0111
	CEV	0.8944 ± 0.0168	0.8309 ± 0.0252

Table 3. Results of the algorithm - correctness

The correctness represents the accuracy of the prediction in the test set, and the learning set correctness represents the accuracy of the model on the learning set. From a global point of view, we can

[5] http://www.uni-marburg.de/fb12/kebi/research/repository/monodata - September 2014
[6] http://archive.ics.uci.edu/ml/ - September 2014
[7] http://www.cs.waikato.ac.nz/ml/weka/datasets.html - September 2014

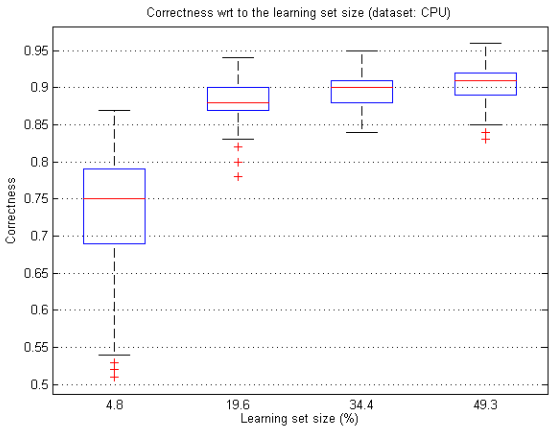

Figure 3. Correctness of the test set wrt LS size - dataset CPU

note that the correctness values are rather good. As expected, the correctness is increasing with the learning set's size. One can note that the learning set correctness is decreasing with the learning set size too. This is because it is much more complicated to have a perfect solution if the number of alternatives in the learning set increases. In particular, the results show that for the dataset CEV, the algorithm does not reach good level of learning set correctness (at least not as high as for the two other one). The reason probably lies in the fact that this specific dataset is much bigger.

On figure 3, we show a boxplot of the evolution of correctness with respect to the learning set size (for the dataset CPU). We can note that a good level of correctness is already reached for a learning size of 20%.

Due to the introduction of the varying parameters for the GA, we did not need to fine tune the parameters anymore. The value of 1600 for the population size has been determined by trial and errors. This looks to be an accurate value. Les us stress that the new method increases the running time. Nevertheless, it remains rather small. For instance, for the CPU dataset, the algorithm runs in about 5 minutes on a Intel i7-2640m with 8go RAM, under Windows 8.1 with an implementation of the algorithm in Java 8. One advantage we have remarked in these first experiments is that the algorithm seems to be less stuck in local optima.

5 Conclusion

In this paper, we have addressed the question of preference elicitation in the context of partial sorting. To the best of our knowledge, this is the first attempt to solve such kind of problems. Furthermore, we have limited the analysis to a specific sorting method namely FlowSort. The approach is based on an extension of a method previously developed for complete sorting. It relies on the use of a dedicated genetic algorithm based on parameters variation. This has been illustrated on three real datasets. The validation was based on a learning set and test set created by a random instantiation of FlowSort. First experiments have shown that the algorithm runs quite fast and leads to good prediction values. A number of research questions are still to be addressed. Among others, we could investigate how an exact method could partly cover the elicitation process (typically the identification of weight values). From an algorithmic point of view, a detailed analysis of the heuristic is still to be done. More precisely, quantitative arguments have to be highlighted in order to confirm the added value of parameters variations. The use of benchmark datasets (that are not linked to a particular method like in this study) will certainly have an impact on the prediction quality. Nevertheless, the existence (or the creation) of such datasets is far from being obvious. Finally, the comparison between different sorting methods will probably lead to identify distinctive features that will be more appropriate to replicate particular categorizations.

REFERENCES

[1] Majid Behzadian, Reza B Kazemzadeh, A Albadvi, and M Aghdasi. Promethee: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*, 200(1):198–215, 2010.

[2] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, 2010.

[3] Jean-Pierre Brans and Bertrand Mareschal. Promethee methods. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 163–186. Springer New York, 2005.

[4] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université libre de Bruxelles, Belgium, 2011.

[5] Bertrand Mareschal, Yves De Smet, and P Nemery. Rank reversal in the promethee ii method: some new results. In *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*, pages 959–963. IEEE, 2008.

[6] Philippe Nemery. *On the use of multicriteria ranking methods in sorting problems*. PhD thesis, PhD Thesis. Université libre de Bruxelles, 2008-2009, 2008.

[7] Philippe Nemery and Claude Lamboray. Flowsort: a flow-based sorting method with limiting or central profiles. *Top*, 16(1):90–113, 2008.

[8] Bernard Roy and Philippe Vincke. Multicriteria analysis: survey and new directions. *European Journal of Operational Research*, 8(3):207–218, 1981.

[9] Olivier Sobrie, Vincent Mousseau, and Marc Pirlot. Learning a majority rule model from large sets of assignment examples. In *Algorithmic Decision Theory*, pages 336–350. Springer, 2013.

[10] Dimitri Van Assche and Yves De Smet. Flowsort parameters elicitation based on classification examples. Technical Report TR/SMG/2014-003, SMG, CoDE, Université Libre de Bruxelles, Brussels, Belgium, June 2014.