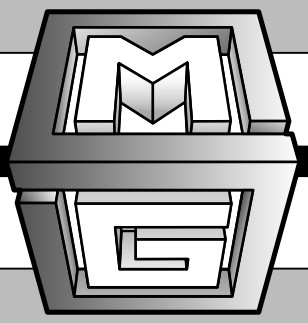

Université Libre de Bruxelles

CoDE - SMG

An extension of PROMETHEE to interval clustering

CoDE-SMG – Technical Report Series

Renaud Sarrazin, Yves De Smet, Jean Rosenfeld

CoDE-SMG – Technical Report Series

Technical Report No.
TR/SMG/2014-009

December 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/2014-009

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.

An extension of PROMETHEE to interval clustering

CoDE-SMG – Technical Report Series

Renaud Sarrazin r.sarrazin@brrc.be

Yves De Smet yves.de.smet@ulb.ac.be

Jean Rosenfeld jean.rosenfeld@ulb.ac.be

Belgian Road Research Centre (BRRC), Wavre, Belgium
CoDE-SMG, Université Libre de Bruxelles, Brussels, Belgium
BEAMS, Université Libre de Bruxelles, Brussels, Belgium

December 2014

An extension of PROMETHEE to interval clustering

R. Sarrazin[a,b,*], Y. De Smet[b], J. Rosenfeld[c]

[a] *MSM division, Belgian Road Research Center, Boulevard de la Woluwe 42, 1200 Brussels, Belgium*
[b] *CoDE-SMG laboratory, Engineering Faculty, Université libre de Bruxelles, Avenue F.D. Roosevelt 50, 1050 Brussels, Belgium*
[c] *BEAMS laboratory, Engineering Faculty, Université libre de Bruxelles, Avenue F.D. Roosevelt 50, 1050 Brussels, Belgium*

Abstract

Multicriteria clustering techniques aim to detect groups of alternatives evaluated on multiple criteria with similar profiles. The preferential partitioning of the dataset allows the decision maker to get a better understanding of the structure of his problem. In this paper, we focus on the particular case of interval clustering. This approach allows us to assign alternatives either in individual or interval clusters. For this purpose, we develop a model based on the PROMETHEE I outranking method and the FlowSort sorting procedure. We evaluate its performances on real-world datasets regarding the convergence, the stability and the quality of the clustering. In particular, we analyse the impact of three update functions and two initialization strategies. This analysis has pointed out some promising results that we underline by comparing the performances of the proposed model with the well-known k-means procedure and the P2CLUST model.

Keywords: Multiple Criteria Analysis, PROMETHEE, FlowSort, Multicriteria Interval Clustering

1. Introduction

We consider decision problems than can be modelled as a set of alternatives evaluated on several conflicting criteria. Facing such situations, researchers

*Corresponding author

Email addresses: r.sarrazin@brrc.be (R. Sarrazin), yves.de.smet@ulb.ac.be (Y. De Smet), jean.rosenfeld@ulb.ac.be (J. Rosenfeld)

usually consider three main types of so-called problematic (Vincke, 1992)

- Choice: choosing a best compromise solution (or a subset of interesting solutions);
- Ranking: ranking the alternatives from the best to the worst one(s) (using either a complete and a partial relation) ;
- Sorting: allocating alternatives into pre-defined categories.

Since a couple of years, a new kind of problematic has emerged in the mulicriteria decision aid community. This is referred as to multicriteria clustering i.e. the detection of groups of alternatives in a multicriteria context.

In data mining, a lot of methods have been proposed to solve the different clustering problems (for instance k-means algorithms, hierarchical methods, support vector machines, etc.). Most of them rely on a distance measure that allows to built groups that are as homogeneous as possible but remain highly heterogeneous between themselves.

In multicriteria contexts, the notion of distance between two alternatives is not really appropriate. This is due to the fact that criteria have to be optimized. Most of the time, the comparison between two alternatives will lead to conclude that the first one is better regarding certain criteria will the second is better for the others points of views. This observation has led to build (binary or valued) preferences relations. To our point of view, these asymmetric relations open new doors for clustering; for instance, the detection of complete or partial relations between the groups.

De Smet and Montano are the first who have investigated the multicriteria clustering problem based on binary preferences (De Smet and Montano Guzmàn, 2004). Their model relies on the definition of a specific distance that takes into account the multicriteria preferential information induced by the comparison of alternatives. Unfortunately, their algorithm was limited to the detection of clusters (nominal clusters). Later, De Smet and Eppe proposed a natural extension of the later work to build relations between the groups (this was futher completed by Eppe et al (Eppe et al., 2014) in the context of valued relations) (De Smet and Eppe, 2009). In this case, no warranty was available regarding the transitivity of the cluster relations or the fact that they were acyclic (even if artificial experiments have shown that such problems were seldom). De Smet et al also developed an exact algorithm to detect a totally ordered clustering (De Smet et al., 2012). Rocha

et al proposed a method for multicriteria clustering in which they distinguish first the clustering approach (which does not integrate the preferences of the decision maker) and then a multicriteria technique to find the relations between the groups (Rocha et al., 2013). More recently, Meyer and Olteanu proposed a formalization of this emerging topic (Meyer and Olteanu, 2013).

In this contribution, we consider a particular multicriteria method called PROMETHEE. It is known for its ease of use, the presence of user-friendly software (Hayez et al., 2012) and a large number of real applications (Behzadian et al., 2010). It is worth noting that a first extension of PROMETHEE for clustering (and sorting) was proposed in 2004 (Figueira et al., 2004). Unfortunately, this preliminary work suffered from some drawbacks like the non respect of the criterion-depency condition (Cailloux et al., 2007). An extension of PROMETHEE II was recently presented (De Smet, 2013). By construction, the relations between the clusters respect the transitivity conditions. In this paper, we study the possibility to further extend PROMETHEE I and to address the problem of interval clustering i.e. the allocations of alternatives not only in individual clusters but also to sets of successive clusters. On the one hand, the proposed extension ensures that relations between the clusters will be acyclic (by construction). On the second hand, the detection of possible interval clusters allows to analyse the problem at two levels: individual clusters indicate clear assignments while interval clusters indicate regions where the allocation is less clear. Such information could lead :

1. to build a new cluster if the number of alternatives belonging to the inverval cluster is important;
2. to detect alternatives that could be considered as outliers (if only a few of them belong to the interval clusters).

The paper is defined as follows. At first, we introduce the main theoretical concepts of the PROMETHEE and FlowSort methods and we present the multicriteria clustering problem. The interval clustering approach is also introduced. Then, we define the proposed model that is based on an extension of the PROMETHEE I method to interval clustering. In the third section, we validate the model on two datasets by evaluating the quality and the stability of the clustering, the respect of the dominance principle and its convergence. Next, we compare the results obtained with this new approach, the formerly P2CLUST model and the k-means procedure.

2. State of the art

2.1. The PROMETHEE methods

The PROMETHEE outranking methods were initiated in the early 80s by Brans and Mareschal (Mareschal et al., 1986; Mareschal and Brans, 2002, 2005; Vincke, 1992). They offer the decision maker a support to solve multicriteria problems by using a valued outranking relation. This relation is based on pairwise comparisons between alternatives and it defines the preference structure of the PROMETHEE method.

Let us consider a set of alternatives $A = \{a_1 \dots a_n\}$ and a set of criteria $F = \{g_1 \dots g_q\}$. Without loss of generality, we suppose that these q criteria have to be maximized. For each criterion g_k, the decision maker evaluates the preference of an alternative a_i over an alternative a_j by measuring the difference of their evaluation on g_k.

$$d_k(a_i, a_j) = g_k(a_i) - g_k(a_j) \tag{1}$$

This pairwise comparison allows the decision maker to quantify how alternative a_i performs on g_k compared to alternative a_j. Then, we use a preference function P_k to transform this value into a preference degree. Depending on the shape of the preference function, the decision maker could define the indifference threshold q_k and the preference threshold p_k for each criterion (cf. Figure 1).

$$P_k(a_i, a_j) = P_k[d_k(a_i, a_j)] \tag{2}$$

$$0 \leq P_k(a_i, a_j) \leq 1 \tag{3}$$

To quantify the global preference of a_i over a_j, we define the notion of preference index $\pi(a_i, a_j)$. It allows us to aggregate all the unicriterion preference $P_k(a_i, a_j)$ by considering the weights ω_k associated to each criterion.

$$\pi(a_i, a_j) = \sum_{k=1}^{q} P_k[d_k(a_i, a_j)] \cdot \omega_k \tag{4}$$

$$\omega_k \geq 0 \text{ and } \sum_{k=1}^{q} \omega_k = 1 \tag{5}$$

The last step of the PROMETHEE methods relies on the calculation of the outranking flows of each action. It allows the decision maker to quantify

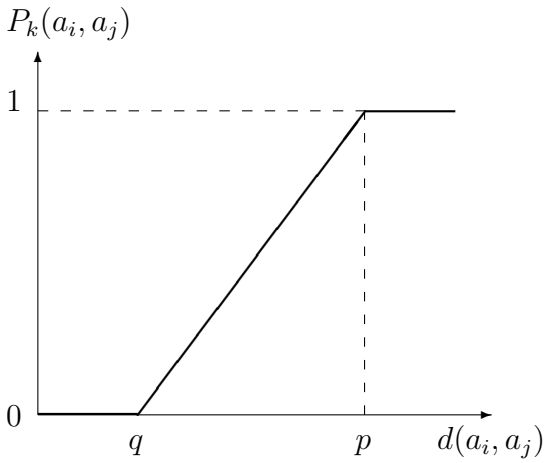

Figure 1: Illustration of a linear preference function in the PROMETHEE methods.

simultaneously how an action a_i is prefered to all the remaining actions x of the set A and how these actions x are prefered to a_i. These two notions are respectively represented by the positive flow ϕ^+ and the negative flow ϕ^- in PROMETHEE I.

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

$$\phi^-(a_i) = \frac{1}{n} \sum_{x \in A} \pi(x, a_i) \tag{7}$$

The positive and negative flows could be combined into the outranking net flow ϕ which is used in PROMETHEE II.

$$\phi(a_i) = \phi^+(a_i) - \phi^-(a_i) \tag{8}$$

Based on the positive and negative flow scores, the PROMETHEE I method generates a partial ranking of the alternatives. In PROMETHEE II, a complete order is generated from the net flow scores of the alternatives.

2.2. The FlowSort method

The FlowSort method was developed by Nemery and Lamboray (Nemery de Bellevaux and Lamboray, 2007) for solving multicriteria sorting problems. This method allows the decision maker to sort the alternatives into categories based on their positive and negative flows. The categories are defined a priori and they remain unchanged during the solving process.

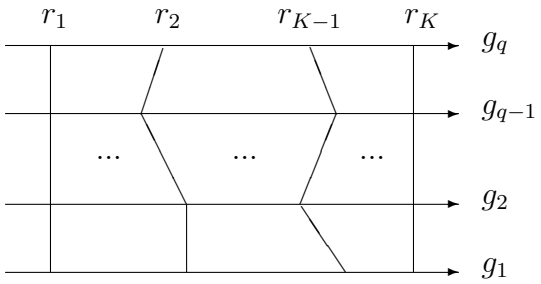

Figure 2: Illustration of central profiles r_j in completely ordered clustering.

Let us consider a set of categories (or clusters) to which the actions will be assigned $\kappa = \{C_1, C_2 \dots C_K\}$. We assume that the K categories are completely ordered such that C_{j+1} is preferred to C_j. In the FlowSort method, the categories could be defined either by one central profile or two limiting profiles. In the following, we will focus on the categories characterized by central profiles. Figure 2 illustrates the central profiles in the FlowSort method for a problem with q criteria and K categories.

Let us denote them $R = \{r_1, r_2 \dots r_K\}$. These reference profiles are representative elements of the category which they belong to and they should respect the dominance principle: r_{j+1} the central profile of C_{j+1} dominates r_j the central profile of C_j (cf. Condition 2.3.1) (Nemery de Bellevaux, 2008).

Condition 2.3.1 $\forall r_h, r_l \in R$ such that $h > l$:
$\forall g_k \in F,\ g_k(r_h) \geq g_k(r_l)$ and $\exists g_x \in F \mid g_x(r_h) > g_x(r_l)$

The fundamental principle of the FlowSort method relies in the association of an alternative a_i to a given category using either the net flow scores of PROMETHEE II or the positive and negative flows of PROMETHEE I. The net flow scores will be used to generate a complete clustering while the positive and negative flows are appropriate in the context of an interval clustering. In practice, we generate for each alternative $a_i \in A$ the combined set $R_i = R \cup \{a_i\}$. Then, the assignment of the alternative to a category is done in two steps. First, we compare its flow to the flows of the central profiles. And then, we assign the alternative to the category whose the profile has the closest flow. With net flow scores, this is formalized by the following condition (Nemery de Bellevaux, 2008).

Condition 2.3.2 $C_\phi(a_i) = C_h$ if: $|\phi_{R_i}(r_h) - \phi_{R_i}(a_i)| = \min_{\forall j} |\phi_{R_i}(r_j) - \phi_{R_i}(a_i)|$

We denote $\delta(A, \kappa)$ the final distribution of the alternatives $a_i \in A$ in the set of categories κ. When the final clustering is of good quality, it produces compact but well-separated categories.

3. Proposed model

Based on the principles of FlowSort and Promethee methods, we have developed the model PCLUST which is an extension of PROMETHEE I for interval clustering. The aim of this model is to solve a multicriteria clustering problem by defining a set of categories κ^* that could be divided in two groups: the principal categories C_i and the interval categories $C_{i,j}$, $\forall i, j \in \{1 \dots K\}$ and $i \neq j$. The principal categories are ordered and respect the dominance principle. While the interval categories $C_{i,j}$ are located "between" the principal categories C_i and C_j. Considering the preference relation of PROMETHEE, it means that the profile $r_{i,j}$ is incomparable with r_i and r_j. The clustering procedure of the PCLUST method is composed of the following steps:

1. Initialization of the central profiles
2. Assignment of the alternatives to the categories
3. Update of the central profiles
4. Repeat the procedure from step 2 until the convergence of the model

In the following, we describe each step of the clustering procedure. The reader who is familiar with the k-means procedure directly see that our approach follows the same mains steps. Nevertheless, two distinctive features have to be highlighted:

a. The allocation is based on a multicriteria sorting method ;
b. The update of profiles have to respect the multicriteria nature of the problem (i.e. the dominance condition) ;

3.1. Initialization of the central profiles

At first, we determine the central profiles either randomly (*Rdm*) or by equidistributing (*Eqd*) the evaluations on every criterion. When initializing the reference profiles randomly, we need to sort the evaluations on every criteria in order to respect the dominance principle between clusters (cf. Condition 2.3.1).

3.2. Assignment of the alternatives to the categories

Next, the assignment of the alternatives to the categories is done with respect to an assignment rule. Let consider an alternative $a_i \in A$ and the set of reference profiles $R = \{r_1 \dots r_K\}$. As in FlowSort, we define the set $R_i = R \cup \{a_i\}$. We compute the preference degrees between the actions of R_i and we calculate the positive and negative flows. Finally, we assign an alternative to a category by following these two conditions:

Condition 3.2.1 $C_{\phi^+}(a_i) = C_h$ if:
$|\phi^+_{R_i}(r_h) - \phi^+_{R_i}(a_i)| = \min_{\forall j} |\phi_{R_i^+}(r_j) - \phi_{R_i^+}(a_i)|$
Condition 3.2.2 $C_{\phi^-}(a_i) = C_l$ if:
$|\phi^-_{R_i}(r_l) - \phi^-_{R_i}(a_i)| = \min_{\forall j} |\phi_{R_i^-}(r_j) - \phi_{R_i^-}(a_i)|$

Based on these conditions, two different categories C_h and C_l could be obtained. In order to assign each alternative to one category, we define the following assignment rule:

Assignment rule 3.2.3 $\forall a_i \in A, \forall h, l \in \{1 \dots K\}$
$\begin{cases} \text{if } C_{\phi^+}(a_i) = C_{\phi^-}(a_i) = C_h, & a_i \in C_h \\ \text{else,} & a_i \in C_{h,l} \end{cases}$

We denote the categories C_h as the principal categories while $C_{h,l}$ are the interval categories ($h \neq l$).

3.3. Update of the central profiles

At the end of each iteration, all the alternatives of the set A are assigned to categories. So, we need to update the reference profile of each category in order to take into consideration this new distribution. In totally ordered clustering, the updated value of the reference profile r_h corresponds to the average value of the evaluations of the alternatives in C_h. However, in interval clustering, the alternatives of the problem could be assigned either in principal or interval categories. Consequently, we could imagine that the updated value of the reference profile r_h would also consider the alternatives in the interval categories $C_{h,j}$ which are related to C_h, $\forall j = \{1 \dots K\}, j \neq h$.

In this paper, we present three functions to update the reference profiles and their respective performances. Considering the principal category C_h, we denote $C_{h,l}$ the interval category that is related to C_h and C_l, and $\{C_{h,j}\}$ the set of all the interval categories that are related to C_h. We denote $\{C_{h,j}\}^h$ the

set of interval categories that are closer to C_h than to C_j, $\forall j \in \{1 \dots K\}, j \neq h$. This notion of closeness refers to the similarity of the PROMETHEE II positive and negative flows, as introduced in the Conditions 3.2.1 and 3.2.2.

*3.3.1. First update function Upd*1

At first, to describe the structure of the first function, we need to distinguish the principal categories that contain at least one alternative and the principal categories that are empty.

(NE) Non-empty principal categories. If the principal category C_h is not empty, the updated value of its reference profile r_h is simply equal to the median value of the evaluations of the alternatives a_i that belong to this category.

$$\begin{gathered} \text{if } |C_h| \neq 0, \forall k \in \{1 \dots q\}, \forall h \in \{1 \dots K\} \\ f_k(r_h) = \frac{1}{|C_h|} \sum_{a_i \in C_h} f_k(a_i) \end{gathered} \tag{9}$$

(E) Empty principal categories. If the principal category C_h is empty, we update the reference profiles differently if the category is extreme (i.e., $h = \{1, K\}$) or not.

(E.e) Extreme principal categories. If the principal category C_h is extreme, the updated value of its reference profile r_h is equal to the median value of the alternatives a_i that belong to the set of interval categories $\{C_{h,j}\}$. If this set $\{C_{h,j}\}$ is empty, the updated value of r_h is equal to a random value comprised in an interval with an upper bound (resp. lower bound) that corresponds to the evaluations of the next (resp. previous) reference profile.

$$\begin{gathered} \text{if } |C_h| = 0, |\{C_{h,j}\}| \neq 0, h \in \{1, K\}, \forall k \in \{1 \dots q\}, \forall j \in \{1 \dots K\} : \\ f_k(r_h) = \frac{1}{|\{C_{h,j}\}|} \sum_{a_i \in \{C_{h,j}\}} f_k(a_i) \end{gathered} \tag{10}$$

$$\begin{gathered} \text{if } |C_h| = 0, |\{C_{h,j}\}| = 0, \forall k \in \{1 \dots q\} : \\ f_k(r_1) = rand \in [0, f_k(r_2)] \\ f_k(r_K) = rand \in [f_k(r_{K-1}), \max_{\forall a_i}(f_k(a_i))] \end{gathered} \tag{11}$$

(E.ne) Non-extreme principal categories. If the principal category C_h is not extreme, the updated value of its reference profile r_h is equal to a random value comprised between the reference profiles of the closest upper category C_{up} and the closest lower category C_{low} that are non-empty.

$$\begin{gathered}\text{if } |C_h| = 0, \forall h \in \{2 \dots K-1\}, \forall k \in \{1 \dots q\} : \\ f_k(r_h) = rand \in [f_k(r_{low}), f_k(r_{up})]\end{gathered} \tag{12}$$

3.3.2. Second update function Upd2

The second update function is very similar to the first one, except for the update of the principal categories that are empty and extreme (E.e).

(E.e) Extreme principal categories. In this configuration, if the principal category C_h is empty while the set of interval categories $\{C_{h,j}\}$ contains at least one alternative, the updated value of the reference profile r_h is calculated only by considering the alternatives from the set of interval categories $\{C_{h,j}\}^h$ that are closer to C_h than to C_j, $\forall j \in \{1 \dots K\}, j \neq h$.

$$\begin{gathered}\text{if } |C_h| = 0, |\{C_{h,j}\}| \neq 0, h \in \{1, K\}, \forall k \in \{1 \dots q\}, \forall j \in \{1 \dots K\} : \\ f_k(r_h) = \frac{1}{|\{C_{h,j}\}^h|} \sum_{a_i \in \{C_{h,j}\}^h} f_k(a_i)\end{gathered} \tag{13}$$

The update procedure for all the others configurations remains the same than for the first function.

3.3.3. Third update function Upd3

Finally, the third update function is similar to the second one, except for the update of the principal categories that are empty and not extreme (E.ne).

(E.ne) Non-extreme principal categories. In this configuration, we distinguish the cases where the the set of interval categories $\{C_{h,j}\}$ is empty or not. If this set is empty, we apply the same update rule than the first function (1.E.ne). If this set is not empty, we apply the same update rule than for the principal categories that are empty and extreme in the second function (2.E.e).

$$\text{if } |C_h| = 0, |\{C_{h,j}\}| \neq 0, \forall k \in \{1 \dots q\}, \forall h, j \in \{1 \dots K\}, j \neq h :$$
$$f_k(r_h) = \frac{1}{|\{C_{h,j}\}^h|} \sum_{a_i \in \{C_{h,j}\}^h} f_k(a_i) \tag{14}$$

The update procedure for all the others configurations remains the same than for the second function.

For each function, the update procedure is concluded by the verification of the order of the reference profiles. In practice, the evaluations of the profiles on every criteria are sorted such that the condition 2.3.1 is verified.

By defining these different functions, we aim to compare the performance of the model during the update procedure. In particular, it will allow us to identify to what extent the use of the preference information from the interval categories would impact the convergence of the model and the quality of the final clustering.

3.4. Repetition of the procedure until convergence of the model

Given that the clustering procedure is iterative, we have to define conditions to stop the model. At first, we define a convergence condition that stops the clustering procedure when the distribution $\delta(A, \kappa)$ remains unchanged during 3 successive iterations. In addition, we define a stopping condition that interrupts the model after 100 iterations without converging.

4. Validation

In this section, we apply the model on two structured datasets from the literature. The first dataset concerns the Environmental Performance Index 2014 (EPI). It is composed of 178 alternatives and 2 criteria (Hsu et al., 2014). The second one is a standard benchmark dataset about CPU evaluation from the UCI repository that had been preprocessed by A.F. Tehrani et. al (Tehrani et al., 2011). It contains 209 alternatives and 6 criteria. For both of these datasets, linear preference functions have been selected for each criterion (cf. Tables 1 and 2). Each test was computed with MATLAB R2013b on a Intel Core i5 CPU 2.40GHz with 4.00GB of RAM.

The aim of the validation process is to compare the three update functions and two initialization strategies of the model. In particular, we would identify if the use of the preferential information from the interval categories improves

Table 1: Parameters of the EPI dataset

n	178
q	2
w	$\{0.4, 0.6\}$
P_k	$\{q_k = 10, p_k = 50\}$

Table 2: Parameters of the CPU dataset

n	209
q	6
w_k	0.167
P_k	$\{q_k = 0.1, p_k = 0.5\}$

the performance of the model. We will evaluate the performances of the PCLUST procedure regarding the quality and the stability of the clustering, the respect of the dominance principle and the convergence of the model.

4.1. Quality of the clustering

At the end of the procedure, we measure the quality of the obtained partition. We assume that a high quality clustering sort the alternatives in the categories such that the inter-distance between the categories is the highest and the intra-distance within each category is the lowest. In the ideal case, the interval categories are at equal distance from the associated principal categories. In addition, we suppose that two clustering distributions with k and $k+1$ clusters would share some similarities. We use a contingency table to illustrate this point.

4.1.1. Quality indicator

In order to represent the relative quality of the clusterings from the computation of each update functions, we have defined a quality indicator that uses the preference information between the alternatives. For a given distribution $\delta(A, \kappa)$ of the alternatives in the categories, we calculate the preference index $\pi(a_i, a_j)$ of each couple of alternatives $a_i, a_j \in A$. In the following, we denote $\pi(a_i, a_j)$ as π_{ij} for simplicity reasons. Then, we evaluate

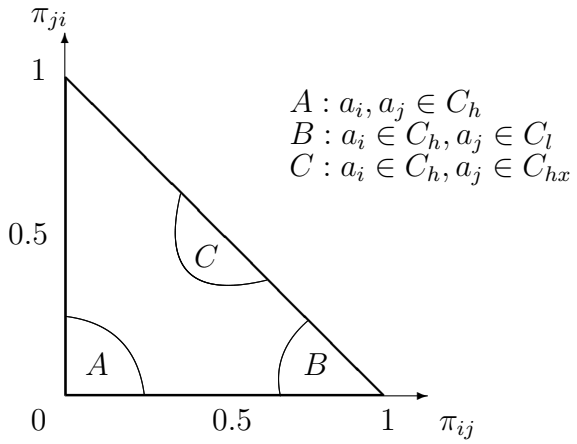

Figure 3: Graphical representation of the values of π_{ij} and π_{ji} for high quality clustering.

the quality index QI_{ij} of the alternatives a_i and a_j as follows:

$$QI_{ij} = \begin{cases} \pi_{ij} + \pi_{ji} & \text{if } \begin{cases} a_i \in C_h \\ a_j \in C_h \end{cases} \\ 1 - \pi_{ij} + \pi_{ji} & \text{if } \begin{cases} a_i \in C_h \\ a_j \in C_l \\ h > l \end{cases} \\ |0.5 - \pi_{ij}| + |0.5 - \pi_{ji}| & \text{if } \begin{cases} a_i \in C_h \\ a_j \in C_{hx} \\ h \neq x \end{cases} \end{cases} \tag{15}$$

Finally, we calculate the global quality index QI of the final clustering by summing the quality indexes QI_{ij}. A graphical representation of the quality index QI is shown on Figure 3. It illustrates the values that should take the preference index π_{ij} and π_{ji} for high quality clustering.

$$QI = \sum_{\substack{\forall a_i, a_j \in A \\ i \neq j}} QI_{ij} \tag{16}$$

When comparing different clusterings, the best distribution of alternatives in the categories would then obtain the lowest global quality index QI. Figures 4 and 5 illustrate the evolution of the clustering quality with the number of categories respectively for the EPI and CPU datasets. The global

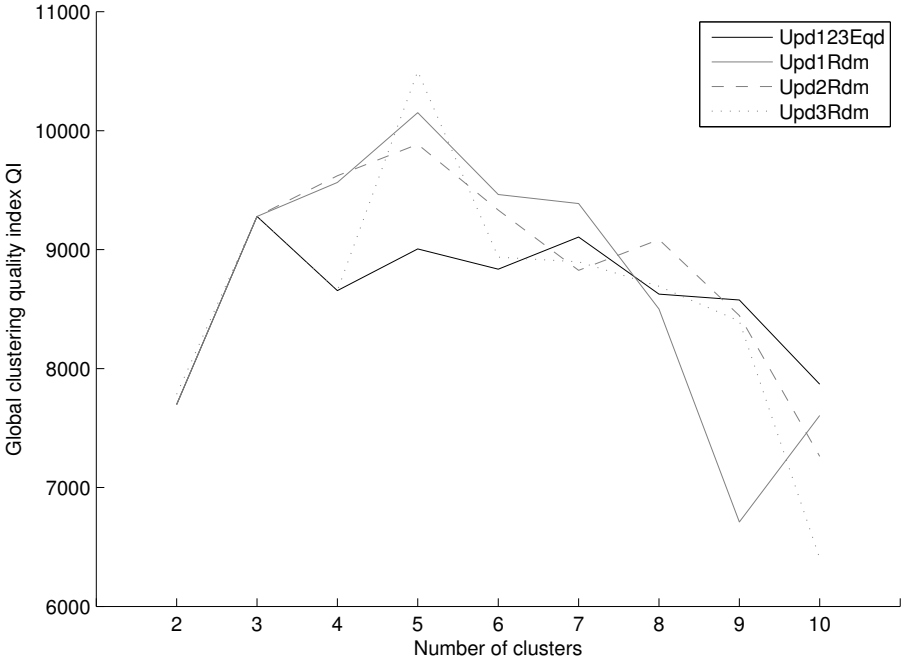

Figure 4: Evolution of the clustering quality with the number of clusters, 30 tests, EPI dataset.

clustering quality index QI is calculated for the 3 update functions and the 2 initialization strategies (i.e., Rdm and Eqd as introduced previously). Each test has been computed 30 times and the average values have been calculated.

We observe on the Figure 4 that the same clustering quality QI is obtained with the 3 update functions and an equidistributed initialization of the reference profiles. However, with a random initialization, the 3 update functions lead to different clustering quality. On average, the 3 update functions generate clustering of similar quality. The function $Upd3Rdm$ obtains the lowest score on the global quality index QI for $k = \{2, 3, 4, 6, 10\}$ while $Upd1Rdm$ and $Upd2Rdm$ obtains the best results respectively for $k = \{2, 3, 8, 9\}$ and $k = \{2, 3, 5, 7\}$. Then, the difference between the functions is low according to the global clustering quality index QI and it is difficult to differentiate them so far.

However, the analysis of the Figure 5 leads to observation slightly in favor of $Upd3Rdm$. Both in random and equidistributed initialization, we observe

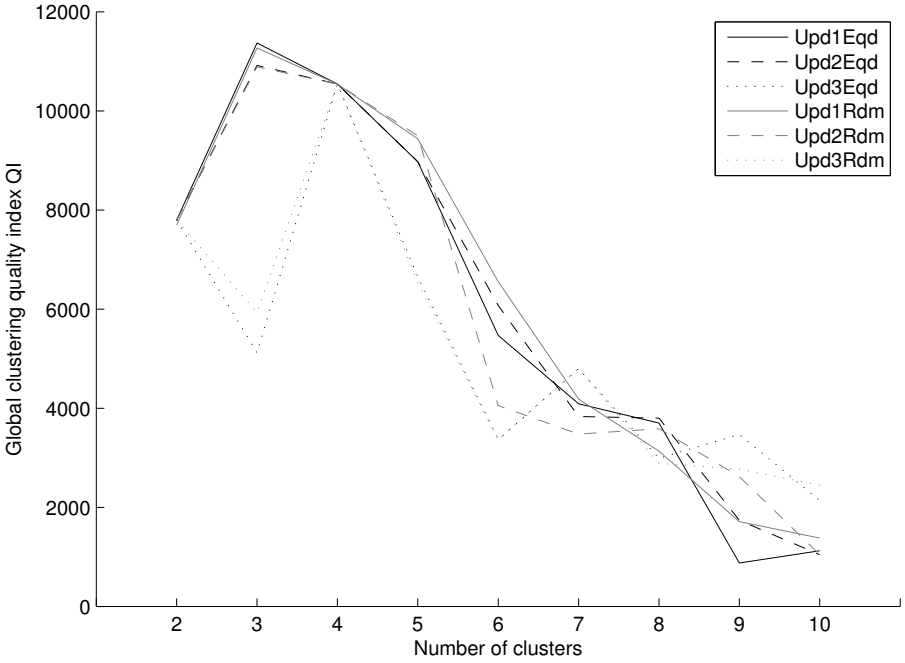

Figure 5: Evolution of the clustering quality with the number of clusters, 30 tests, CPU dataset.

that the third update function obtains lower or equal scores on the global quality index QI for $k = \{2, 3, 4, 5, 6, 8\}$. However, except for $k = \{3, 5\}$, the difference on the clustering quality remains low between the three update functions and the two initialization strategies. In addtion, we clearly see on the Figure 5 that there is a strong influence of the number of categories on the clustering quality.

4.1.2. Contingency table

If the model performs well, it should generate high quality clustering independently of the number of categories. Additionnaly, we could imagine that the distribution of the alternatives between a clustering with k categories and a clustering with $k + 1$ categories would share some similarities. In particular, we assume that the creation of an additional category would not transform completely the structure of the initial clustering.

To verify this assumption, we generate the contingency table with the

Table 3: Contingency table of two clusterings with $k = 2$ and $k = 3$ categories, first update function, random initialization, EPI dataset

	$\lvert C_1\rvert$	$\lvert C_{12}\rvert$	$\lvert C_2\rvert$	$\lvert C_{23}\rvert$	$\lvert C_3\rvert$	$\sum$
$\lvert C_1\rvert$	56	7	7	0	0	70
$\lvert C_{12}\rvert$	0	2	50	6	0	58
$\lvert C_2\rvert$	0	0	0	13	37	50
$\sum$	56	9	57	19	37	**178**

distribution of the alternatives for a clustering with respectively 2 and 3 categories for the EPI dataset (cf. Table 3). This table allows us to compare the distribution of alternatives in the categories for the first update function and a random initialization of the central profiles. The use of the second and third update functions during the clustering procedure leads to the same type of results.

We clearly observe in the Table 3 that the alternatives assigned to the categories C_1, C_{12} and C_2 of the distribution $\kappa_{k=2}$ are mainly located in the principal categories C_1, C_2 and C_3 of the distribution $\kappa_{k=3}$. In addition, we see that the spread of the alternatives is limited to maximum two principal categories and one interval category. Consequently, when adding a principal category to the clustering procedure, the new distribution of the alternatives seems consistent compared to the previous one. This observation is also illustrated on the visual representation of the distributions $\kappa_{k=2}$ and $\kappa_{k=3}$ on the Figure 6. It shows the consistency and the quality of the clustering aside from the number of categories, the type of update procedure or the initialization conditions.

4.2. Convergence

The analysis of the convergence of the model is a good indicator to measure the performance of each update function and the influence of the initialization strategies. We have run the model 100 times for each update function (Upd) and each initialization stragety (Eqd and Rdm). Table 4 indicates the average number of iterations that are required to observe a convergence of the model and the standard deviation.

For the EPI dataset, we clearly see that the influence of the update functions on the convergence is not significant. For the CPU dataset, the first

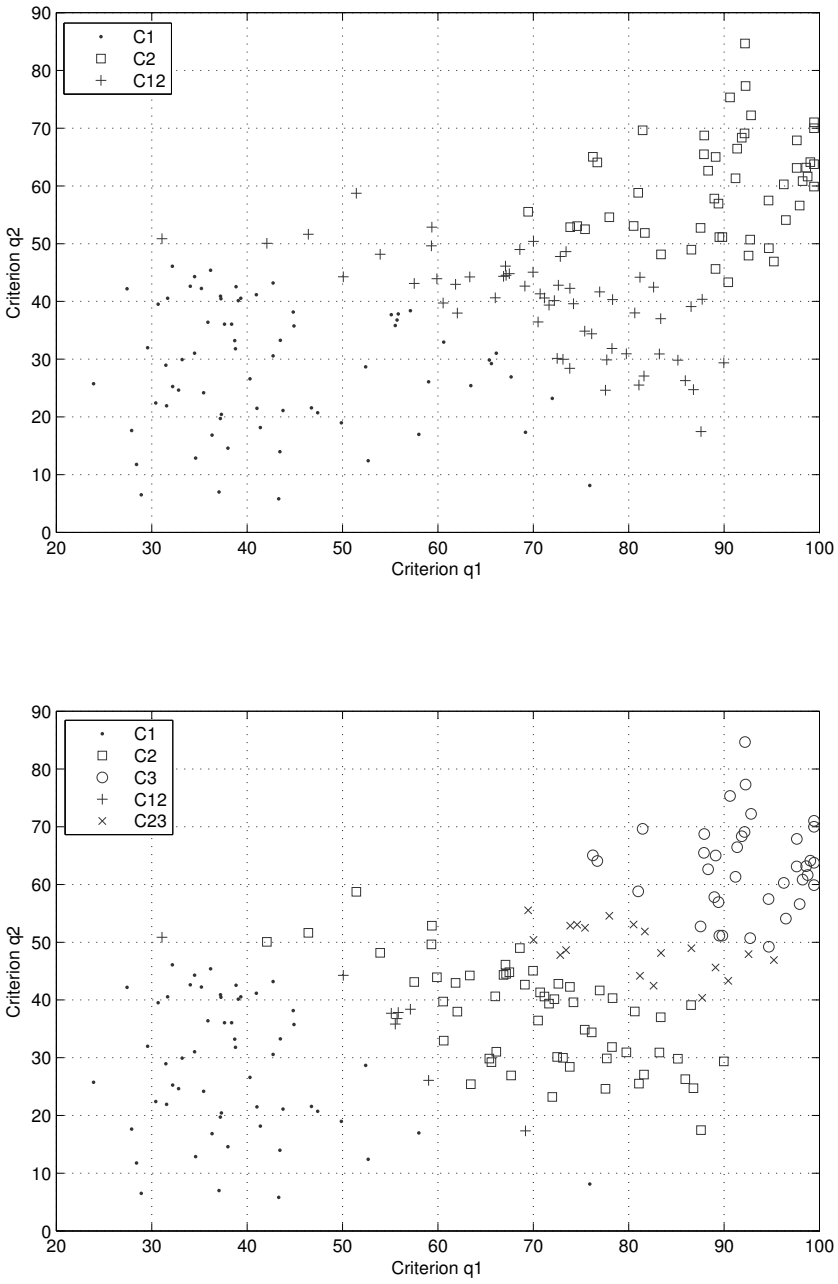

Figure 6: Visual representation of the clustering for $k = 2$ and $k = 3$ ($Upd1Rdm$), EPI dataset.

Table 4: Average number of iterations to converge (i_{tot}) and standard deviation (std), 100 runs, EPI ($k = 4$) and CPU ($k = 4$) datasets.

	EPI		CPU	
	i_{tot}	std	i_{tot}	std
$Upd1Eqd$	17	0	16.81	4.65
$Upd2Eqd$	17	0	19.91	6.39
$Upd3Eqd$	17	0	25.31	0.49
$Upd1Rdm$	11.09	5.74	14.58	4.79
$Upd2Rdm$	9.50	4.81	15.19	6.03
$Upd3Rdm$	10.11	4.62	17.62	9.94

update function is the fastest while the third one requires more iterations to converge. Concerning the initialization strategy, it seems that the use of random or equidistributed initialization has a stonger influence on the convergence. Both for the EPI and CPU datasets, the random initialization of the reference profiles leads to a fastest convergence compared to the equidistribution of the evaluations when initializing the clustering procedure.

Figure 7 illustrate the evolution of the distribution during the clustering procedure for all the update functions and initialization strategies. Each test has been computed 100 times such that the difference between two successive distributions corresponds to an average value. We observe almost no difference between the three update functions, while the influence of the initialization strategy is slightly more important. Nevertheless, the convergence of the model is good for every clustering procedure.

4.3. Stability

We assume that a clustering procedure is stable when the final distribution of the alternatives in the categories does not vary much from one run of the model to another. In other words, the stability of the algorithm is represented by its ability to generate the same clustering structure for a given dataset.

Then, we measure the stability of the model on the EPI and CPU datasets. For each update function and initial condition, we run the model 100 times and we identify which distribution $\delta(A, \kappa)$ is the most represented at the end of each clustering procedure. We define the stability $\mathcal{S}$ as the occurrence of the most represented distribution after 100 runs of the model. The results of

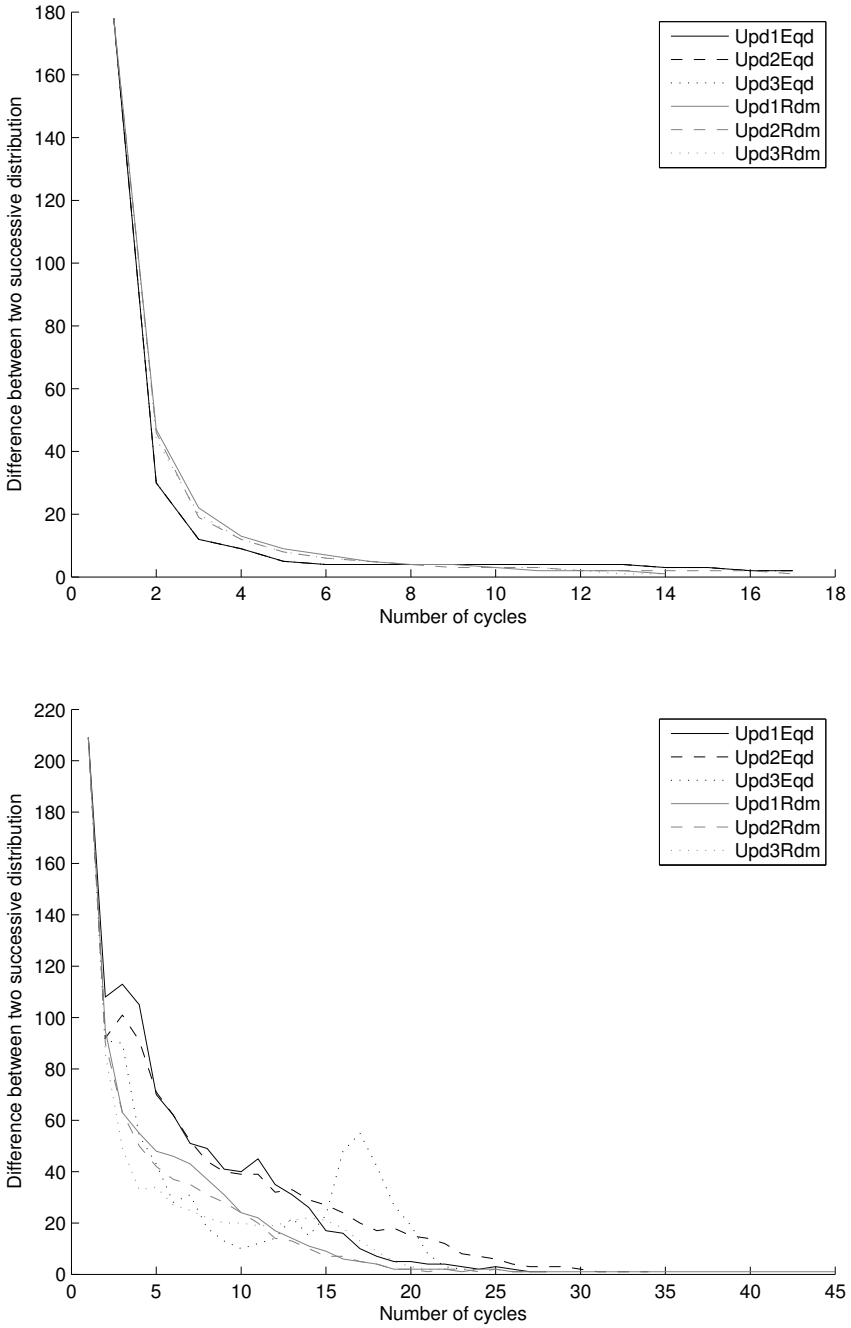

Figure 7: Difference between two successive distribution during the clustering procedure, 100 tests, EPI (top) and CPU (bottom) datasets.

Table 5: Stability of the clustering $\mathcal{S}$ (%), 100 runs, EPI ($k = 4$) and CPU ($k = 4$) datasets.

	$\mathcal{S}$	
	(EPI)	(CPU)
$Upd1Eqd$	100	93
$Upd2Eqd$	100	99
$Upd3Eqd$	100	100
$Upd1Rdm$	39	88
$Upd2Rdm$	37	92
$Upd3Rdm$	41	95

Table 6: Proportion of distribution $\delta_i(A, \kappa)$ after 100 runs (in %), procedure $Upd1Rdm$, EPI dataset.

$\delta_1(A, \kappa)$	3%
$\delta_2(A, \kappa)$	39%
$\delta_3(A, \kappa)$	30%
$\delta_4(A, \kappa)$	2%
$\delta_5(A, \kappa)$	3%
$\delta_6(A, \kappa)$	3%
$\delta_7(A, \kappa)$	15%
$\delta_8(A, \kappa)$	1%
$\delta_9(A, \kappa)$	4%

these measures of stability are presented in the Table 5.

We clearly notice that the stability of the clustering is very high when the reference profiles are initiated from an equidistribution of the evaluations on every criteria. On the contrary, the stability drops bellow 50% for the EPI dataset when initializing the reference profiles randomly. It remains between 88% and 95% for the CPU dataset. The analysis of the results for the EPI dataset has pointed out that 9 different distributions were generated when applying the random initialization and the first update function (11 with $Upd2Rdm$, 9 with $Upd3Rdm$). Table 6 shows their representation in the set of distributions after 100 runs of the model. We can see that 2 of these 9 distributions represent 69% of then entire set.

Table 7: Stability of the clustering $\mathcal{S}$ (%) with 2% of allowed error, 100 runs, EPI ($k = 4$) dataset.

	$\mathcal{S}_{2\%}$ (EPI)
$Upd1Eqd$	100
$Upd2Eqd$	100
$Upd3Eqd$	100
$Upd1Rdm$	69
$Upd2Rdm$	70
$Upd3Rdm$	71

However, a comparative analysis of the distributions $\delta_2(A, \kappa)$ and $\delta_3(A, \kappa)$ shows that only 2 alternatives among 178 are affected to a different category. Consequently, if these distributions are not strictly equivalent, they are highly similar. Table 7 shows the stability of the clustering for the EPI dataset when allowing 2% of error between two distributions. It illustrates clearly that the stability of the model on the EPI dataset remains acceptable with a random initialization of the reference profiles.

4.4. Respect of the dominance principle

Finally, we must verify that the model satisfy the dominance principle. Considering that the categories are ordered such that C_{j+1} is prefered to C_j, the respect of the dominance principle implies that the central profile of C_{j+1} dominates the central profiles of C_j (Condition 2.3.1). A corollary of this condition is that the alternatives that are assigned to the category C_{j+1} cannot be dominated by the alternatives that are assigned to C_j.
Dominance tests were then conducted on the EPI and CPU datasets. Any dominance issue was revealed and the model was validated regarding the respect of the dominance principle.

5. Comparison with existing procedures

Once the PCLUST model has been defined and tested on several indicators, we could compare its performance with the well known k-means procedure (Bottou and Bengio, 1995; Bradley and Fayyad, 1998; Kanungo et al., 2002)

and the P2CLUST model. In the following, we analyse the results of the models on the EPI and CPU datasets introduced previously.

At first, let us compare the clustering distribution of each model with the PROMETHEE II ranking. For simplicity reasons, we only consider the $Upd1Eqd$ procedure for the PCLUST model and the equidistributed initialization strategy for the P2CLUST model. The analysis is done on the CPU dataset. We observe on Table 8 that the first alternative of the PII ranking is assigned to the best cluster C_4 with the PCLUST and P2CLUST models, while it is located in the category C_3 with the k-means procedure. Overall, the structure of the k-means distribution weakly respects the PROMETHEE ranking. This observation comes as no surprise given that the k-means model is not suitable for ordered clustering. Consequently, it underlines that even if the PCLUST model seems to share some similarities with the k-means procedure, these two methods generate highly different clustering distributions. In addition, when focusing on the principal categories in the Table 8, we observe that the PCLUST distribution stricly respect the PROMETHEE II ranking. Indeed, there is no overlap in the PROMETHEE II ranking between alternatives assigned to different principal categories. With the P2CLUST model, we detect some ranking issues (e.g. alternatives ranked at position 77 and 78, 121 and 122). When analyzing the complete PROMETHEE II ranking, we report 26 ranking issues over the 209 alternatives (12%) with the P2CLUST model. Consequently, the use of interval clustering approach allows the decision maker to generate a clustering distribution that is closer to the PROMETHEE II ranking than when using a totally ordered clustering model.

The comparison of the global clustering quality index scores also underlines the better performance of the PCLUST model. Figure 8 shows that the values of QI are significantly worst when using the P2CLUST model rather than PCLUST on the EPI dataset. Similar results are observed on the CPU dataset. Moreover, we observe that the quality of the k-means and P2CLUST clusterings decreases when the number of clusters increases, while it is improved with the PCLUST model.

Concerning the convergence of PCLUST and P2CLUST, Table 9 shows the performance of the two models on both datasets. We observe that the P2CLUST model converges slightly faster than PCLUST when comparing the number of iterations. However, the gain remains moderate even when comparing the calculation times. On the EPI dataset, the PCLUST model requires between 50 and 76 seconds to compute 100 runs while the P2CLUST

Table 8: Comparison of the clustering distribution of the PCLUST, P2CLUST and k-means models with the PROMETHEE II ranking, 100 runs, CPU dataset ($k = 4$).

PII		$\delta(A, \kappa)$		
rank	ϕ	P2CLUST	PCLUST	k-means
1	0,664	4	4	3
2	0,495	4	4	4
3	0,491	4	4	4
4	0,486	4	4	4
5	0,446	4	4	4
6	0,421	4	4	3
7	0,397	4	4	3
8	0,368	4	4	3
9	0,359	4	4	3
10	0,326	4	4	4
11	0,300	304	4	3
12	0,300	304	4	3
13	0,289	304	4	3
14	0,255	304	4	2
15	0,237	3	4	3
16	0,193	3	4	2
...		...	...	...
71	-0,001	3	4	2
72	-0,001	3	4	2
73	-0,001	203	4	1
74	-0,002	203	4	2
75	-0,005	203	4	2
76	-0,005	203	4	2
77	-0,006	2	3	2
78	-0,009	203	4	2
79	-0,012	103	3	1
...		...	...	...
120	-0,066	102	1	1
121	-0,068	102	1	1
122	-0,069	2	2	1
123	-0,069	1	2	1
124	-0,069	102	1	1
...		...	...	...
204	-0,102	1	1	1
205	-0,103	1	1	1
206	-0,106	1	1	1
207	-0,106	1	1	1
208	-0,107	1	1	1
209	-0,109	1	1	1

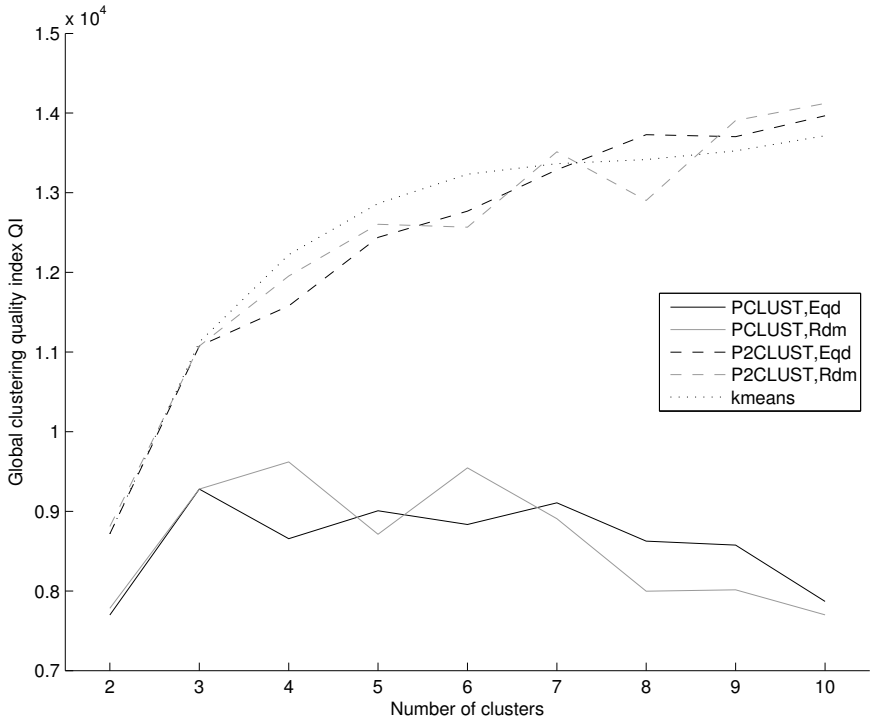

Figure 8: Evolution of the clustering quality with the number of clusters, 30 tests, EPI dataset. Comparison of the models PCLUST, P2CLUST and k-means.

Table 9: Average number of iterations to converge (i_{tot}), standard deviation (std) and total calculation time (t_{100} in seconds), methods PCLUST and P2CLUST, 100 runs, EPI ($k = 4$) and CPU ($k = 4$) datasets.

	EPI			CPU		
	i_{tot}	std	$t_{100}(s)$	i_{tot}	std	$t_{100}(s)$
PCLUST						
$Upd1Eqd$	17	0	66.88	16.81	4.65	185.91
$Upd2Eqd$	17	0	66.47	19.91	6.39	215.05
$Upd3Eqd$	17	0	70.89	25.31	0.49	276.17
$Upd1Rdm$	11.09	5.74	56.40	14.58	4.79	158.60
$Upd2Rdm$	9.50	4.81	53.73	15.19	6.03	168.11
$Upd3Rdm$	10.11	4.62	57.35	17.62	9.94	226.07
P2CLUST						
Eqd	8	0	44.39	13.52	0.91	178.38
Rdm	5.95	2.49	39.36	9.01	2.80	145.92

method needs 40 to 45 seconds. On the CPU dataset, the gain is about 1.15 to 1.7 times greater for the P2CLUST model.

Finally, concerning the stability of the clustering, the Table 10 shows that the P2CLUST model performs well on the EPI and CPU datasets with equidistributed initialization strategy while it obtains bad to average results when generating the initial reference profiles randomly. The PCLUST model obtains slightly better results on the two datasets. With 2% of allowed error between different distributions, the stability of PCLUST on the EPI dataset increases up to 70% in the worst case while it remains around 23% for P2CLUST. On the CPU dataset, the stability remains unchanged and in favor of PCLUST.

Consequently, the comparison of the PCLUST model with the k-means procedure and the P2CLUST method has shown interesting results. The PCLUST model obtains great performance regarding the stability and the quality of the final clustering. Nevertheless, the performance results on the convergence indicator are weakly in favor of P2CLUST.

Table 10: Stability of the clustering $\mathcal{S}$ and $\mathcal{S}_{2\%}$ (%), methods PCLUST and P2CLUST, 100 runs, EPI ($k = 4$) and CPU ($k = 4$) datasets.

	$\mathcal{S}$		$\mathcal{S}_{2\%}$	
	(EPI)	(CPU)	(EPI)	(CPU)
PCLUST				
$Upd1Eqd$	100	93	100	94
$Upd2Eqd$	100	99	100	99
$Upd3Eqd$	100	100	100	100
$Upd1Rdm$	39	88	69	88
$Upd2Rdm$	37	92	70	96
$Upd3Rdm$	41	95	71	96
P2CLUST				
Eqd	100	100	100	100
Rdm	19	61	23	61

6. Conclusions

In this paper, we developed an extension of PROMETHEE I to interval clustering and we tested the performance of the proposed model on several indicators. The originality of this approach relies the use of the preferential information among alternatives given by the PROMETHEE I method to solve the multicriteria problem of interval clustering.

The proposed model uses the PROMETHEE positive and negative flow scores to assign the alternatives to the corresponding categories. We solve the problem by strictly using the preferential information among alternatives on every criteria. Then, it limitates the loss of information during the solving process and it allows the decision maker to generate clustering with higher quality.

Moreover, the performance analysis of the model on the EPI and CPU datasets has shown interesting results. In particular, the stability and the quality of the final clustering are particularly good with the PCLUST model. Concerning the clustering quality, the comparison of the PCLUST model with the P2CLUST approach and the k-means procedure pointed out that the proposed model improves the quality index score from 20% with 4 categories up to 275% with 8 categories on the CPU dataset. Moreover, the PCLUST model obtains on average a better stability than the P2CLUST approach. As

regards the convergence of the model, PCLUST obtains acceptable results even if it requires slightly higher calculation time in comparison with the P2CLUST model.

Furthermore, the analysis of three different update procedures for the PCLUST model pointed out the limited interest of using the preferential information from the interval categories. In addition, we studied the impact of the equidistributed and random initialization strategies on the performance of the model. It emerges from this analysis that the equidistributed initialization of the reference profiles led to more stable clustering while the random initialization allows the model to converge faster.

To conclude, the PCLUST model developed in this paper had led to promising results. The use of the PROMETHEE I flow scores in the context of interval clustering constitute an added-value in the field of research of multicriteria clustering. Moreover, the analysis of the performance of the model on real-world datasets are encouraging. The comparison of the proposed model with the well-known k-means procedure and the formerly developed P2CLUST method has underlined a strong interest in using such an approach to characterize complex multicriteria clustering problems.

Acknowledgement

This study is supported by the Operational Department of Economy, Employment and Research of the Walloon Region (Belgium), under the First DoCA financing program [number 1017209].

References

Behzadian, M., Kazemzadeh, R., Albadvi, A., Aghdasi, M., 2010. PROMETHEE: A comprehensive literature review on methodologies and applications. European Journal of Operational Research 200 (1), 198 – 215. URL http://dx.doi.org/10.1016/j.ejor.2009.01.021

Bottou, L., Bengio, Y., 1995. Convergence Properties of the k-Means Algorithms. In: Advances in Neural Information Processing Systems 7. MIT Press, pp. 585–592.

Bradley, P. S., Fayyad, U. M., 1998. Refining Initial Points for k-Means Clustering. Morgan kaufmann, pp. 91–99.

Cailloux, O., Lamboray, C., Nemery, P., et al., 2007. A taxonomy of clustering procedures. In: Proceedings of the 66th Meeting of the European Working Group on MCDA.

De Smet, Y., 2013. P2CLUST: an extension of PROMETHEE II for ordered clustering. In: 2013 IEEE International Conference on Industrial Engineering and Engineering Management. Bangkok, Thailand.

De Smet, Y., Eppe, S., 2009. Relational multicriteria clustering: The case of binary outranking matrices. In: Ehrgott, M. e. a. (Ed.), Evolutionary Multi-Criterion Optimization. Fifth international conference, EMO 2009. Proceedings. Vol. 5467 of Lecture Notes in Computer Science. Springer Berlin, pp. 380–392.

De Smet, Y., Montano Guzmàn, L., 2004. Towards multicriteria clustering: An extension of the k-means algorithm. European Journal of Operational Research 158 (2), 390 – 398, methodological Foundations of Multi-Criteria Decision Making.
URL http://dx.doi.org/10.1016/j.ejor.2003.06.012

De Smet, Y., Nemery, P., Selvaraj, R., 2012. An exact algorithm for the multicriteria ordered clustering problem. Omega 40 (6), 861 – 869, special Issue on Forecasting in Management Science.
URL http://dx.doi.org/10.1016/j.omega.2012.01.007

Eppe, S., Roland, J., De Smet, Y., 2014. On the use of valued action profiles for relational multi-criteria clustering. International Journal of Multicriteria Decision Making.

Figueira, J., De Smet, Y., Brans, J.-P., 2004. MCDA methods for sorting and clustering problems: PROMETHEE TRI and PROMETHEE CLUSTER. Research report, SMG-ULB.

Hayez, Q., De Smet, Y., Bonney, J., 2012. D-Sight: A new decision making software to address multi-criteria problems. Int. J. of Decision Support System Technology.

Hsu, A., Emerson, J., Levy, M., de Sherbinin, A., Johnson, L., Malik, O., Schwartz, J., Jaiteh, M., 2014. The 2014 environmental performance index. Tech. rep., Yale Center for Environmental Law & Policy.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., Wu, A. Y., Member, S., Member, S., 2002. An efficient k-means clustering algorithm: Analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 881–892.

Mareschal, B., Brans, J.-P., 2002. PROMETHEE : Une méthodologie d'aide à la décision en présence de critères multiples. Collection "Statistique et Mathématiques Appliquées". Editions de l'Université de Bruxelles, Paris, France.

Mareschal, B., Brans, J.-P., 2005. PROMETHEE methods. In: Multicriteria Decision Analysis: State of the Art Surveys. International Series in Operations Research and Management Science. Springer-Verlag, pp. 163–196.

Mareschal, B., Brans, J.-P., Vincke, P., 1986. How to select and how to rank projects: the PROMETHEE method. European Journal of Operational Research 24, 228–238.

Meyer, P., Olteanu, A.-L., 2013. Formalizing and solving the problem of clustering in {MCDA}. European Journal of Operational Research 227 (3), 494 – 502.
URL http://dx.doi.org/10.1016/j.ejor.2013.01.016

Nemery de Bellevaux, P., November 2008. On the use of multicriteria ranking methods in sorting problems. Ph.D. thesis, Université libre de Bruxelles, Brussels, Belgium.

Nemery de Bellevaux, P., Lamboray, C., 2007. FlowSort : a flow-based sorting method with limiting and central profiles. TOP (Official Journal of the Spanish Society of Statistics and Operations Research) 16 (1), 90–113.

Rocha, C., Dias, L. C., Dimas, I., 2013. Multicriteria classification with unknown categories: A clusteringsorting approach and an application to conflict management. Journal of Multi-Criteria Decision Analysis 20 (1-2), 13–27.
URL http://dx.doi.org/10.1002/mcda.1476

Tehrani, A. F., Cheng, W., Hüllermeier, E., July 2011. Choquistic regression: generalizing logistic regression using the Choquet integral. In: Galichet,

S., Montero, J., Mauris, G. (Eds.), Proceedings of the 7th Conference of the European Society for Fuzzy Logic and Technology. Atlantis Press, Aix-Les-Bains, France, pp. 868–875.

Vincke, P., 1992. Multicriteria Decision-Aid. J. Wiley, New York.