PCLUST: An extension of PROMETHEE to interval clustering

Renaud Sarrazin\textsuperscript{1,2}, Yves De Smet\textsuperscript{2}, Jean Rosenfeld\textsuperscript{3}

\textsuperscript{1} MSM division, Belgian Road Research Centre, Brussels, Belgium
\textsuperscript{2} CoDE-SMG laboratory, Université libre de Bruxelles, Brussels, Belgium
\textsuperscript{3} BEAMS laboratory, Université libre de Bruxelles, Brussels, Belgium

2\textsuperscript{nd} International MCDA Workshop on PROMETHEE: Research and Case Studies – Extensions and Theoretical Developments
Research question

Dealing with a problem of multicriteria clustering

Aim of this research:
- Consider the multicriteria nature of the problem
- Construct interval clusters (i.e., in partial order)
**PCLUST: An extension of PROMETHEE to interval clustering**

**Context**

Problematic of multicriteria clustering
Non-relational, relational and ordered clustering

[A-L Olteanu, 2013]
PCLUST: An extension of PROMETHEE to interval clustering

Context

Non-relational clustering: no criteria-dependency

Many methods exist to solve clustering problems (e.g. $k$-means algorithms)
Most of them rely on a **distance measure**!
- Minimize the inner-distance of each group
- Maximize the inter-distance between groups
- Not really appropriate in multicriteria contexts

Use of the **preference relations** between alternatives!

**Illustration of the $k$-means algorithm:**

(Wikipedia)
**Context**

Relational and ordered clustering: criteria-dependency
Consider the additional information given by the criteria

In this contribution, we focus on **multicriteria ordered clustering**
In particular, we address the problem of interval clustering (i.e. partial order)
Strong interest in using such an approach

Illustrative example:

- 2 criteria
- 80 alternatives located in 4 areas of the space
- 2 principal clusters required

Solution
- **Completely ordered clustering**
- **Interval clustering**
FlowSort method

Developed for solving problems of multicriteria sorting

Let consider:

- \( A = \{a_1, \ldots, a_n\} \): set of alternatives
- \( F = \{g_1, \ldots, g_q\} \): set of criteria
- \( \kappa = \{C_1, \ldots, C_K\} \): set of clusters
- \( R = \{r_1, \ldots, r_K\} \): set of corresponding central profiles
- \( R_i = R \cup \{a_i\} \)

Assignment rule

\[ C_\phi(a_i) = C_h \text{ if: } |\phi_{R_i}(r_h) - \phi_{R_i}(a_i)| = \min_{\forall j} |\phi_{R_i}(r_j) - \phi_{R_i}(a_i)| \]
PCLUST: An extension of PROMETHEE to interval clustering

PCLUST model

Enrichment of the \textit{k-means} procedure with the \textit{FlowSort} assignment rule
Adaptation of the assignment rule to interval clustering

\textbf{Pseudo code}

1. Initialization of the central profiles
2. Assignment of each alternatives to the categories
3. Update of the central profiles
4. Repeat until convergence of the model
**PCLUST: An extension of PROMETHEE to interval clustering**

**PCLUST model**

1. **Initialization**
   - Randomly
   - Equidistribution of the evaluations

2. **Assignment rule**

   \[
   C_{\phi^+}(a_i) = C_h \quad \text{if} \quad |\phi_{R_i}^+(r_h) - \phi_{R_i}^+(a_i)| = \min_{\forall j} |\phi_{R_i}^+(r_j) - \phi_{R_i}^+(a_i)|
   \]

   \[
   C_{\phi^-}(a_i) = C_l \quad \text{if} \quad |\phi_{R_i}^-(r_h) - \phi_{R_i}^-(a_i)| = \min_{\forall j} |\phi_{R_i}^-(r_j) - \phi_{R_i}^-(a_i)|
   \]

   \[\forall a_i \in A, \forall h, l \in \{1 \ldots K\} : \begin{cases} 
   \text{if } C_{\phi^+}(a_i) = C_{\phi^-}(a_i) = C_h : a_i \in C_h \\
   \text{else} : a_i \in C_{h,l}
   \end{cases}\]
PCLUST: An extension of PROMETHEE to interval clustering

PCLUST model

3. **Update of the central profiles // First function Upd1**
   - Non-empty principal categories - median value
   - Empty principal categories
     - Extreme categories
       - Interval - median value
       - No interval - bounded random value
     - Non-extreme categories - bounded random value

\[
C_1 = \{\emptyset\} \\
C_i \neq \{\emptyset\} \\
C_i = \{\emptyset\} \cup K = \{\emptyset\}
\]
PCLUST model

3. Update of the central profiles // Second function Upd2
   - Non-empty principal categories - median value
   - Empty principal categories
     - Extreme categories
       - Interval -
       - No interval - bounded random value
   - Non-extreme categories - bounded random value

\[ C_1 = \{\emptyset\} \]
PCLUST: An extension of PROMETHEE to interval clustering

PCLUST model

3. Update of the central profiles // Second function Upd2

- Non-empty principal categories - median value
- Empty principal categories
  - Extreme categories
    - Interval - median value of the closest alternatives
    - No interval - bounded random value
  - Non-extreme categories - bounded random value

\[ C_1 = \{\emptyset\} \]
PCLUST: An extension of PROMETHEE to interval clustering

PCLUST model

3. **Update of the central profiles // Third function Upd3**
   - Non-empty principal categories - median value
   - Empty principal categories
     - Extreme categories
       - Interval - median value of the closest alternatives
       - No interval - bounded random value
     - Non-extreme categories

\[ C_i = \{\emptyset\} \]
PCLUST: An extension of PROMETHEE to interval clustering

PCLUST model

3. Update of the central profiles // Third function Upd3

- Non-empty principal categories - median value
- Empty principal categories
  - Extreme categories
    - Interval - median value of the closest alternatives
    - No interval - bounded random value
- Non-extreme categories
  - Interval - median value of the closest alternatives
  - No interval - bounded random value

\[ C_i = \emptyset \]
PCLUST: An extension of PROMETHEE to interval clustering

Validation

Evaluation of the update functions and initialization procedures
Comparison with existing procedures ($k$-means and P2CLUST)

Two structured datasets
- Environmental Performance Index (EPI 2014)
- CPU evaluation (UCI repository)

Table 1: Parameters of the EPI dataset

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>178</td>
</tr>
<tr>
<td>$q$</td>
<td>2</td>
</tr>
<tr>
<td>$w$</td>
<td>${0.4, 0.6}$</td>
</tr>
<tr>
<td>$P_k$</td>
<td>${q_k = 10, p_k = 50}$</td>
</tr>
</tbody>
</table>

Table 2: Parameters of the CPU dataset

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>209</td>
</tr>
<tr>
<td>$q$</td>
<td>6</td>
</tr>
<tr>
<td>$w_k$</td>
<td>0.167</td>
</tr>
<tr>
<td>$P_k$</td>
<td>${q_k = 0.1, p_k = 0.5}$</td>
</tr>
</tbody>
</table>
Validation - Quality

Let denote \( \pi_{ij} \) the preference index \( \pi(a_i, a_j) \).
Definition of the quality index \( Q_{IJ_{ij}} \):

\[
Q_{IJ_{ij}} = \begin{cases} 
\pi_{ij} + \pi_{ji} & \text{if } \begin{cases} a_i \in C_h \\
a_j \in C_h \end{cases} \\
1 - \pi_{ij} + \pi_{ji} & \text{if } \begin{cases} a_i \in C_h \\
a_j \in C_l \\
h > l \end{cases} \\
|0.5 - \pi_{ij}| + |0.5 - \pi_{ji}| & \text{if } \begin{cases} a_i \in C_h \\
a_j \in C_{hx} \\
h \neq x \end{cases}
\end{cases}
\]

The lower is \( Q_{IJ_{ij}} \), the better is the quality of the final clustering distribution.
Validation - Quality

![Graph showing evolution of clustering quality with number of clusters]

**Figure**: Evolution of the clustering quality with the number of clusters, 30 tests, EPI dataset.
Validation - Quality

Figure: Evolution of the clustering quality with the number of clusters, 30 tests, CPU dataset
Validation - Quality

Contingency table of two clustering distributions with k = 2 and k = 3 categories (Upd1Rdm, EPI dataset)

|      | $|C_1|$ | $|C_{12}|$ | $|C_2|$ | $|C_{23}|$ | $|C_3|$ | $\Sigma$ |
|------|-------|---------|-------|---------|-------|--------|
| $|C_1|$ | 56    | 7       | 7     | 0       | 0     | 70     |
| $|C_{12}|$| 0     | 2       | 50    | 6       | 0     | 58     |
| $|C_2|$ | 0     | 0       | 0     | 13      | 37    | 50     |
| $\Sigma$| 56    | 9       | 57    | 19      | 37    | 178    |

Limited spread of the alternatives when adding a cluster
Assignment to principal clusters in priority
Same results were observed with Upd2 and Upd3
Validation - Convergence

Calculation of the average number of iterations to converge \( (i_{tot}) \)
Datasets EPI (k=4) and CPU (k=4), 100 runs

<table>
<thead>
<tr>
<th></th>
<th>EPI</th>
<th></th>
<th>CPU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( i_{tot} )</td>
<td>std</td>
<td>( i_{tot} )</td>
<td>std</td>
</tr>
<tr>
<td>( U_{pd1Eqd} )</td>
<td>17 0</td>
<td></td>
<td>16.81 4.65</td>
<td></td>
</tr>
<tr>
<td>( U_{pd2Eqd} )</td>
<td>17 0</td>
<td></td>
<td>19.91 6.39</td>
<td></td>
</tr>
<tr>
<td>( U_{pd3Eqd} )</td>
<td>17 0</td>
<td></td>
<td>25.31 0.49</td>
<td></td>
</tr>
<tr>
<td>( U_{pd1Rdm} )</td>
<td>11.09 5.74</td>
<td></td>
<td>14.58 4.79</td>
<td></td>
</tr>
<tr>
<td>( U_{pd2Rdm} )</td>
<td>9.50 4.81</td>
<td></td>
<td>15.19 6.03</td>
<td></td>
</tr>
<tr>
<td>( U_{pd3Rdm} )</td>
<td>10.11 4.62</td>
<td></td>
<td>17.62 9.94</td>
<td></td>
</tr>
</tbody>
</table>

Influence of the update functions is **not significant**.
**Random initialization** of the profiles has a stronger influence on the convergence.
Validation - Stability

Calculation of the stability of the clustering $S$ after 100 runs (%), EPI (k=4) and CPU (k=4) Proportion of distribution $\delta_i(A, \kappa)$ after 100 runs (%), EPI (k=4), Upd1Rdm

<table>
<thead>
<tr>
<th></th>
<th>EPI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upd1Eqd</td>
<td>100</td>
<td>93</td>
</tr>
<tr>
<td>Upd2Eqd</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>Upd3Eqd</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Upd1Rdm</td>
<td>39</td>
<td>88</td>
</tr>
<tr>
<td>Upd2Rdm</td>
<td>37</td>
<td>92</td>
</tr>
<tr>
<td>Upd3Rdm</td>
<td>41</td>
<td>95</td>
</tr>
</tbody>
</table>

$\delta_1(A, \kappa)$ 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%

Stability **globally good**, except for EPI dataset in random initialization. But the distributions $\delta_2(A, \kappa)$ and $\delta_3(A, \kappa)$ are **very similar** in that case (error < 2%).
Comparison with existing procedures - Quality

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

Calculation of average number of iterations to converge ($i_{tot}$), standard deviation ($std$) and total calculation time $t_{100}$ (in seconds). Datasets EPI (k=4) and CPU (k=4), 100 runs.

<table>
<thead>
<tr>
<th></th>
<th>EPI</th>
<th></th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$i_{tot}$</td>
<td>std</td>
<td>$t_{100}(s)$</td>
</tr>
<tr>
<td>PCLUST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upd1Eqd</td>
<td>17</td>
<td>0</td>
<td>66.88</td>
</tr>
<tr>
<td>Upd2Eqd</td>
<td>17</td>
<td>0</td>
<td>66.47</td>
</tr>
<tr>
<td>Upd3Eqd</td>
<td>17</td>
<td>0</td>
<td>70.89</td>
</tr>
<tr>
<td>Upd1Rdm</td>
<td>11.09</td>
<td>5.74</td>
<td>56.40</td>
</tr>
<tr>
<td>Upd2Rdm</td>
<td>9.50</td>
<td>4.81</td>
<td>53.73</td>
</tr>
<tr>
<td>Upd3Rdm</td>
<td>10.11</td>
<td>4.62</td>
<td>57.35</td>
</tr>
<tr>
<td>P2CLUST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eqd</td>
<td>8</td>
<td>0</td>
<td>44.39</td>
</tr>
<tr>
<td>Rdm</td>
<td>5.95</td>
<td>2.49</td>
<td>39.36</td>
</tr>
</tbody>
</table>

P2CLUST converges **slightly faster** when comparing the iterations. **Gain remains moderate** even when comparing the calculation times.
PCLUST: An extension of PROMETHEE to interval clustering

Comparison with existing procedures - Stability

Calculation of the stability of the clustering $S$ (%) and the stability allowing 2% of error $S_{2\%}$ (%), EPI (k=4) and CPU (k=4) datasets, 100 runs

<table>
<thead>
<tr>
<th></th>
<th>$S$ (EPI)</th>
<th>$S$ (CPU)</th>
<th>$S_{2%}$ (EPI)</th>
<th>$S_{2%}$ (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLUST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Upd1Eqd$</td>
<td>100</td>
<td>93</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>$Upd2Eqd$</td>
<td>100</td>
<td>99</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>$Upd3Eqd$</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$Upd1Rdm$</td>
<td>39</td>
<td>88</td>
<td>69</td>
<td>88</td>
</tr>
<tr>
<td>$Upd2Rdm$</td>
<td>37</td>
<td>92</td>
<td>70</td>
<td>96</td>
</tr>
<tr>
<td>$Upd3Rdm$</td>
<td>41</td>
<td>95</td>
<td>71</td>
<td>96</td>
</tr>
<tr>
<td>P2CLUST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Eqd$</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$Rdm$</td>
<td>19</td>
<td>61</td>
<td>23</td>
<td>61</td>
</tr>
</tbody>
</table>

Results are significantly better with the PCLUST model in random initialization. Results are similar with the equidistributed initialization strategy.
**Conclusions**

First extension of PROMETHEE to interval clustering.

Validation of the model on real-world datasets underlines interesting results. Stability and quality of the clustering are particularly good. Interval clustering allows to generate higher quality clustering distributions. Acceptable convergence of the model.

Limited interest of using preferential information from interval clusters. Equidistributed initialization leads to more stable clustering. Random initialization allows the model to converge faster.