

Université de Tahri Mohammed – Béchar
Faculté de Math et informatique

Module : AMCD

Rapport de TP sur la méthode Electre II

Par : Mendja Wadie
Sous la supervision de : Ziani S

Année universitaire 2021/2022

Dans ce TP nous allons appliquer la méthode d'Electre II sur l'exemple que vous nous avez donné qui est comme suit :

	g1	g2	g3	g4	g5	g6	g7
W	0.078	0.118	0.157	0.314	0.235	0.039	0.059
a1	1	2	1	5	2	2	4
a2	3	5	3	5	3	3	3
a3	3	5	3	5	3	2	2
a4	1	2	2	5	1	1	1
a5	1	1	3	5	4	1	5

J'ai créé mon programme avec le langage java, la méthode principale (main) contient ce qui suit :

```

162 static String[][] doubleToMatrix(String[][] matrix) {
163     String[][] stringMatrix = new String[matrix.length][matrix[0].length];
164     for (int i = 0; i < matrix.length; i++) {
165         for (int j = 0; j < matrix[i].length; j++) {
166             stringMatrix[i][j] = matrix[i][j] + "";
167         }
168     }
169     return stringMatrix;
170 }
171
172 public static void main(String[] args) {
173     double[][] tableDePerformance = {
174         { 1, 2, 1, 5, 2, 2, 4 },
175         { 3, 5, 3, 5, 3, 3, 3 },
176         { 3, 5, 3, 5, 3, 2, 2 },
177         { 1, 2, 2, 5, 1, 1, 1 },
178         { 1, 1, 3, 5, 4, 1, 5 }
179     };
180     double[] poids = { 0.078, 0.118, 0.157, 0.314, 0.235, 0.039, 0.059 };
181     double seuilConc = 0.75, seuilDisc = 0.25;
182     afficher(tableDePerformance, "Table de performance :");
183     double[][] matriceDeConc = concordance(tableDePerformance, poids);
184     afficher(matriceDeConc, "Matrice de concordance :");
185     double[][] matriceDeDisc = discordance(tableDePerformance);
186     afficher(matriceDeDisc, "Matrice de discordance :");
187     double[][] matriceCred = matriceCredibilite(matriceDeConc, matriceDeDisc, seuilConc, seuilDisc);
188     afficher(matriceCred, "Matrice de credibilite :");
189     ArrayList<String> ranking = rank(matriceCred);
190     System.out.println("Ranking : " + ranking);
191 }

```

Annotations in the image:

- Matrice de Performance**: Points to the `tableDePerformance` array initialization.
- Vecteur des poids**: Points to the `poids` array initialization.
- Seuils de concordance et de discordance**: Points to the `seuilConc` and `seuilDisc` variable declarations.

Après sa on l'appelle de la méthode **concordance()** ; qui va prendre deux paramètres :

- La table de performance
- Et le vecteur de poids

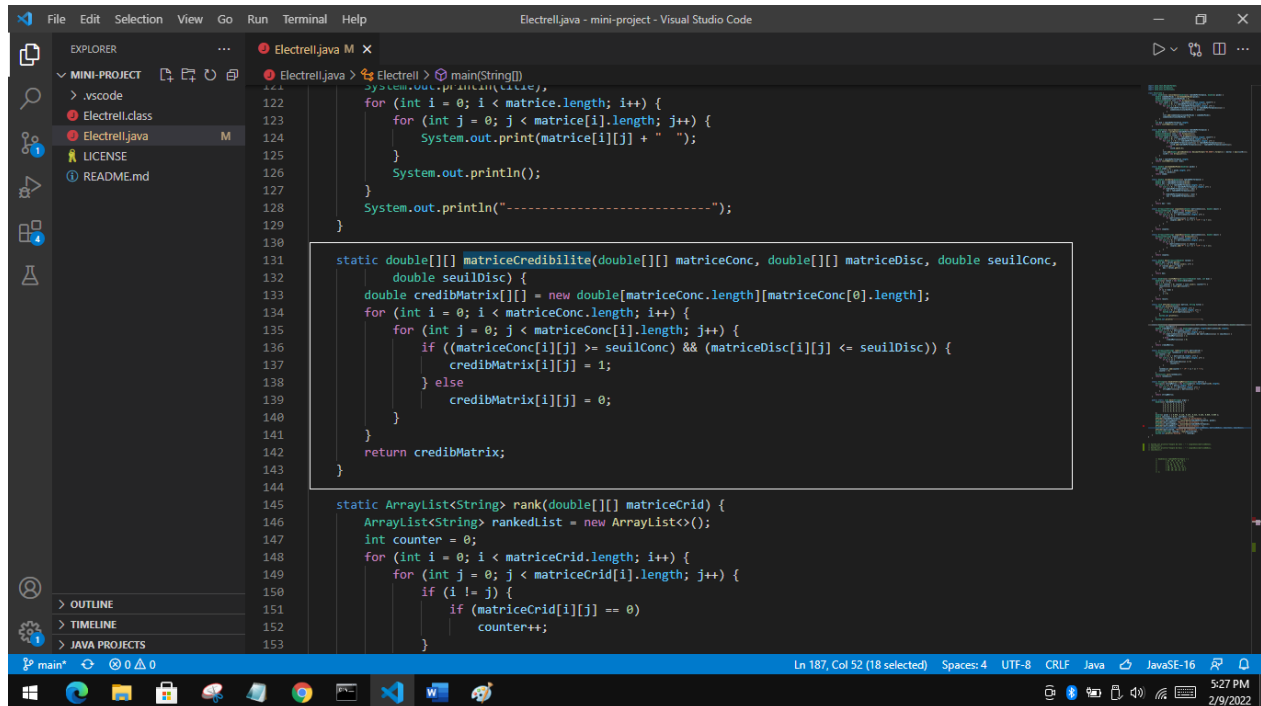
Et va retourner une matrice carrée (dans ce cas-là 5x5) ressemble à ceci :

```
1 import java.text.DecimalFormat;
2 import java.util.ArrayList;
3 import java.util.Collections;
4
5 class ElectreII {
6     static double[][] concordance(double[][] tableDePerformance, double[] poids) {
7         double sommeDesPoids = calcSommeDesPoids(poids);
8         double sommeConditionnalDesPoids = 0;
9         ArrayList<Double> list = new ArrayList<>();
10        for (int count = 0; count < tableDePerformance.length; count++) {
11            for (int i = 0; i < tableDePerformance.length; i++) {
12                for (int j = 0; j < tableDePerformance[i].length; j++) {
13                    if (tableDePerformance[count][j] >= tableDePerformance[i][j]) {
14                        sommeConditionnalDesPoids += poids[j];
15                    }
16                }
17                list.add(sommeConditionnalDesPoids / sommeDesPoids);
18                sommeConditionnalDesPoids = 0;
19            }
20        }
21        int dim = tableDePerformance.length;
22        return listToMatrix(list, dim);
23    }
24
25    static double[][] discordance(double[][] tableDePerformance) {
26        double delta = calcDelta(tableDePerformance);
27        ArrayList<Double> list = new ArrayList<>();
28        ArrayList<Double> list0 = new ArrayList<>();
29        for (int count = 0; count < tableDePerformance.length; count++) {
30            for (int i = 0; i < tableDePerformance.length; i++) {
31                for (int j = 0; j < tableDePerformance[i].length; j++) {
32                    if (tableDePerformance[count][j] <= tableDePerformance[i][j]) {
33                        list0.add(tableDePerformance[i][j] - tableDePerformance[count][j]);
34                    } else {
35                        list0.add(0.0);
36                    }
37                }
38                list.add(Double.parseDouble(new DecimalFormat("###.####").format((1 / delta) * max(list0))));
39                list0 = new ArrayList<>();
40            }
41        }
42        int dim = tableDePerformance.length;
43        return listToMatrix(list, dim);
44    }
45
46    static double calcSommeDesPoids(double[] poids) {
47        double somme = 0;
48        for (int i = 0; i < poids.length; i++) {
49            somme += poids[i];
50        }
51        return somme;
52    }
53}
```

Même chose avec la méthode qui calcule la matrice de **discordance()** :

```
195 }
196
197 int dim = tableDePerformance.length;
198 return listToMatrix(list, dim);
199 }
200
201 static double[][] discordance(double[][] tableDePerformance) {
202     double delta = calcDelta(tableDePerformance);
203     ArrayList<Double> list = new ArrayList<>();
204     ArrayList<Double> list0 = new ArrayList<>();
205     for (int count = 0; count < tableDePerformance.length; count++) {
206         for (int i = 0; i < tableDePerformance.length; i++) {
207             for (int j = 0; j < tableDePerformance[i].length; j++) {
208                 if (tableDePerformance[count][j] <= tableDePerformance[i][j]) {
209                     list0.add(tableDePerformance[i][j] - tableDePerformance[count][j]);
210                 } else {
211                     list0.add(0.0);
212                 }
213             }
214             list.add(Double.parseDouble(new DecimalFormat("###.####").format((1 / delta) * max(list0))));
215             list0 = new ArrayList<>();
216         }
217     }
218     int dim = tableDePerformance.length;
219     return listToMatrix(list, dim);
220 }
221
222 static double calcSommeDesPoids(double[] poids) {
223     double somme = 0;
224     for (int i = 0; i < poids.length; i++) {
225         somme += poids[i];
226     }
227     return somme;
228 }
229 }
```

Calcul de la matrice de crédibilité : la méthode ***matriceCredibilite()*** ; qui va prendre quatre paramètres, les deux matrice (conc et disc) et sont seuils :



```
File Edit Selection View Go Run Terminal Help
Electrell.java - mini-project - Visual Studio Code

EXPLORER
MINI-PROJECT
  .vscode
  Electrell.class
  Electrell.java
  LICENSE
  README.md

OUTLINE
TIMELINE
JAVA PROJECTS

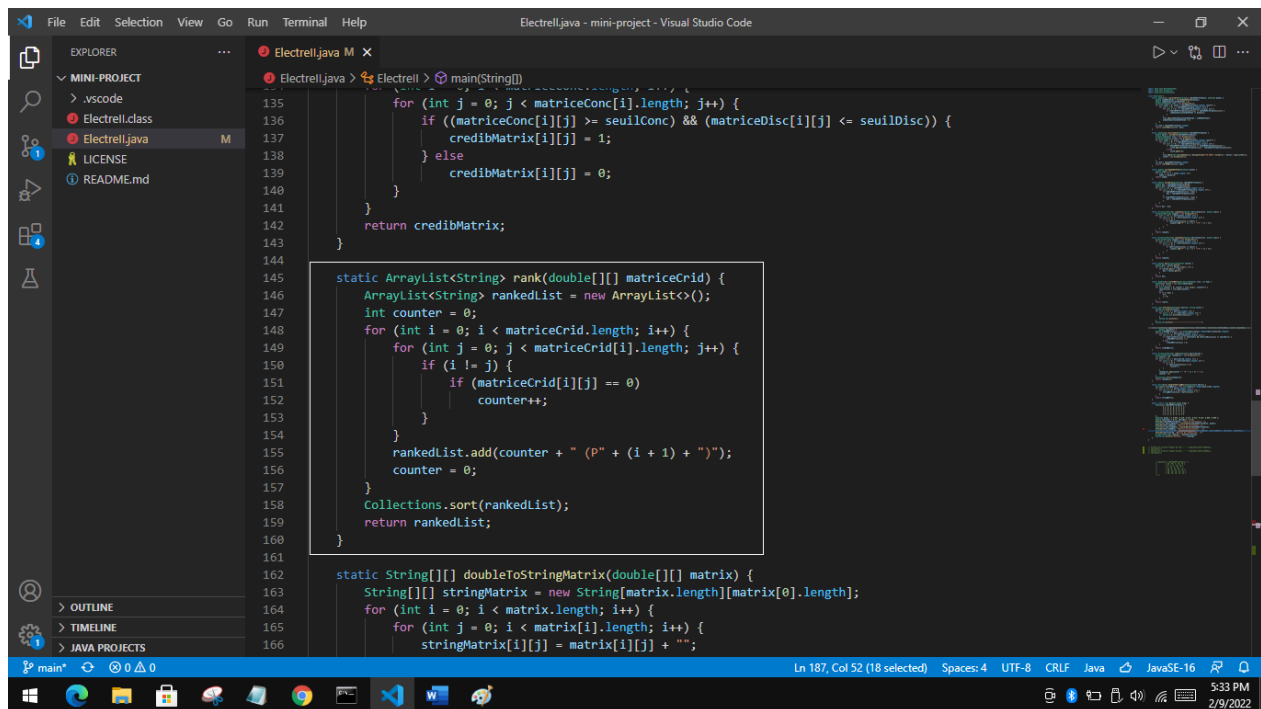
Electrell.java
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153

static double[][] matriceCredibilite(double[][] matriceConc, double[][] matriceDisc, double seuilConc,
double seuilDisc) {
    double credibMatrix[][] = new double[matriceConc.length][matriceConc[0].length];
    for (int i = 0; i < matriceConc.length; i++) {
        for (int j = 0; j < matriceConc[i].length; j++) {
            if ((matriceConc[i][j] >= seuilConc) && (matriceDisc[i][j] <= seuilDisc)) {
                credibMatrix[i][j] = 1;
            } else {
                credibMatrix[i][j] = 0;
            }
        }
    }
    return credibMatrix;
}

static ArrayList<String> rank(double[][] matriceCrid) {
    ArrayList<String> rankedList = new ArrayList<>();
    int counter = 0;
    for (int i = 0; i < matriceCrid.length; i++) {
        for (int j = 0; j < matriceCrid[i].length; j++) {
            if (i != j) {
                if (matriceCrid[i][j] == 0)
                    counter++;
            }
        }
    }
    rankedList.add(counter + " (P" + (i + 1) + ")");
    counter = 0;
    Collections.sort(rankedList);
    return rankedList;
}

static String[][] doubleToStringMatrix(double[][] matrix) {
    String[][] stringMatrix = new String[matrix.length][matrix[0].length];
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            stringMatrix[i][j] = matrix[i][j] + " ";
        }
    }
    return stringMatrix;
}
```

Ranking : la méthode ***rank()*** ; prendre la matrice de crédibilité comme paramètre et retourner le vecteur de ranking :



```
File Edit Selection View Go Run Terminal Help
Electrell.java - mini-project - Visual Studio Code

EXPLORER
MINI-PROJECT
  .vscode
  Electrell.class
  Electrell.java
  LICENSE
  README.md

OUTLINE
TIMELINE
JAVA PROJECTS

Electrell.java
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166

static double[][] matriceCredibilite(double[][] matriceConc, double[][] matriceDisc, double seuilConc,
double seuilDisc) {
    double credibMatrix[][] = new double[matriceConc.length][matriceConc[0].length];
    for (int i = 0; i < matriceConc.length; i++) {
        for (int j = 0; j < matriceConc[i].length; j++) {
            if ((matriceConc[i][j] >= seuilConc) && (matriceDisc[i][j] <= seuilDisc)) {
                credibMatrix[i][j] = 1;
            } else {
                credibMatrix[i][j] = 0;
            }
        }
    }
    return credibMatrix;
}

static ArrayList<String> rank(double[][] matriceCrid) {
    ArrayList<String> rankedList = new ArrayList<>();
    int counter = 0;
    for (int i = 0; i < matriceCrid.length; i++) {
        for (int j = 0; j < matriceCrid[i].length; j++) {
            if (i != j) {
                if (matriceCrid[i][j] == 0)
                    counter++;
            }
        }
    }
    rankedList.add(counter + " (P" + (i + 1) + ")");
    counter = 0;
    Collections.sort(rankedList);
    return rankedList;
}

static String[][] doubleToStringMatrix(double[][] matrix) {
    String[][] stringMatrix = new String[matrix.length][matrix[0].length];
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[i].length; j++) {
            stringMatrix[i][j] = matrix[i][j] + " ";
        }
    }
    return stringMatrix;
}
```

L'exécution de program :

```
C:\WINDOWS\system32\cmd.exe
D:\2MIAD\AMCD\mini-project>javac ElectreII.java
D:\2MIAD\AMCD\mini-project>java ElectreII
Table de performance :
1.0 2.0 1.0 5.0 2.0 2.0 4.0 135
3.0 5.0 3.0 5.0 3.0 3.0 3.0 136
3.0 5.0 3.0 5.0 3.0 2.0 2.0 137
1.0 2.0 2.0 5.0 1.0 1.0 1.0 138
1.0 1.0 3.0 5.0 4.0 1.0 5.0 139
-----
Matrice de concordance :
1.0 0.373 0.412 0.843 0.549 140
0.9410000000000001 1.0 1.0 1.0 0.7060000000000001 141
0.9410000000000001 0.902 1.0 1.0 0.7060000000000001
0.667 0.314 0.314 1.0 0.549 142
0.843 0.7649999999999999 0.7649999999999999 0.8819999999999999 1.0 143
-----
Matrice de discordance :
0.0 0.75 0.75 0.25 0.5 144
0.25 0.0 0.0 0.0 0.5 145
0.5 0.25 0.0 0.0 0.75 146
0.75 0.75 0.75 0.0 1.0 147
0.25 1.0 1.0 0.25 0.0 148
-----
Matrice de credibilite :
1.0 0.0 0.0 1.0 0.0 149
1.0 1.0 1.0 1.0 0.0 150
0.0 1.0 1.0 1.0 0.0 151
0.0 0.0 0.0 1.0 0.0 152
1.0 0.0 0.0 1.0 1.0 153
-----
Ranking : [1 (P2), 2 (P3), 2 (P5), 3 (P1), 4 (P4)]
D:\2MIAD\AMCD\mini-project>
```

Vous pouvez vérifier tout mon code source dans mon compte GitHub ici :

<https://github.com/wadiemendja/electre-II>

Mon email : wadiemendja@gmail.com

Merci 😊