University of Luxembourg

FACULTY OF SCIENCE, TECHNOLOGY AND MEDICINE

UNIVERSITÉ DU LUXEMBOURG

Algorithmic Decision Theory

Problem 23

Moad HANI

ID: 0190054306

moad.hani.001@student.uni.lu

July 2, 2020

Outline:

- Contextualization
- Analysis

  - Bipolar Outranking (Gamma and not Gamma)
  - Global vs relative rating
  - Best ranking method among (Copeland, Netflows and Kohler rankings)
  - K-Best ranking methods

– Conclusion in Q8.

Appendices:

I- Listing the content of Digraph after moving my two files namely per_Tab23.py and HistoricalData_23.py inside the Directory using sftp protocol.

II- Dev environement: CentOS VM - Azure Cloud : Due to many issues working with my local environement (Windows 10), I prefered to move to work with Linux in a test env instead.

III- Installing packages (Azure-CLI, Bastion, R tools, nose, graphviz and cython.

IV- Sftp – Migrating my project from local machine to the VM.

V- The Python Script afferent to project 23.

References:

1. https://orbilu.uni.lu/browse?type=authorulg&rpp=20&value=Bisdorff%2C+Raymond+50000801
2. Digraph3 repo (from where I downloaded the source code afferent to Digraph3) https://github.com/rbisdorff/Digraph3

Contextualization :

All operations on data were done using Digraph3 Python3 package, which implements decision aid algorithms useful in the field of Algorithmic Decision Theory and more specifically in outranking based Multiple Criteria Decision Aid (MCDA).

1 Illustrate the content of the given *perf_Tab.py* performance tableau by best showing objectives, criteria, decision actions and performance table. If needed, write adequate python code.

Let's discover our data … According to the documentation:

1. A **name** attribute, holding usually the actual name of the stored instance that was used to create the instance;
2. A collection of digraph nodes called **actions** (decision actions): an ordered dictionary of nodes with at least a 'name' attribute;
3. An **order** attribute containing the number of graph nodes (length of the actions dictionary) automatically added by the object constructor;
4. A logical characteristic **valuationdomain**, a dictionary with three decimal entries: the minimum (-1.0, means certainly false), the median (0.0, means missing information) and the maximum characteristic value (+1.0, means certainly true);
5. The digraph **relation** : a double dictionary indexed by an oriented pair of actions (nodes) and carrying a decimal characteristic value in the range of the previous valuation domain;
6. Its associated **gamma function** : a dictionary containing the direct successors, respectively predecessors of each action, automatically added by the object constructor;
7. Its associated **notGamma function** : a dictionary containing the actions that are not direct successors respectively predecessors of each action, automatically added by the object constructor.

   And aditionally (and to serve the outranking purpose) we have :

8. a coherent family of **criteria**: a dictionary of criteria functions used for measuring the performance of each potential decision action with respect to the preference dimension captured by each criterion,

9. the **evaluations**: a dictionary of performance evaluations for each decision action or alternative on each criterion function

So, Let’s first look at best ranked decisions from a global multi-objectives compromise point of view. We just create a performance table and then plot a heat map with ranked decisions as follows:

```
_t = Performance Tableau ( 'per_Tab 23 ' )
_t . showHTMLPerformanceHeatmap ( Correlations = True , ndigits = 0 , colorLevels = 10)
```

Listing 1: Python code for Global Point of View Heatmap

And with show_ functions we can always take a look at all attributes. At first glance, we see that weights are equally distributed over the same decision objective. This is because all the data set is randomly generated with the ‘equiobjective’ argument.

```
>>> tglobal.showObjectives()
*------ decision objectives -------"
Eco: Economical aspect
   ec01 criterion of objective Eco 35
   ec07 criterion of objective Eco 35
   ec13 criterion of objective Eco 35
  Total weight: 105.00 (3 criteria)

Soc: Societal aspect
   so02 criterion of objective Soc 21
   so04 criterion of objective Soc 21
   so05 criterion of objective Soc 21
   so06 criterion of objective Soc 21
   so08 criterion of objective Soc 21
  Total weight: 105.00 (5 criteria)

Env: Environmental aspect
   en03 criterion of objective Env 15
   en09 criterion of objective Env 15
   en10 criterion of objective Env 15
   en11 criterion of objective Env 15
   en12 criterion of objective Env 15
   en14 criterion of objective Env 15
   en15 criterion of objective Env 15
  Total weight: 105.00 (7 criteria)
```

```
>>> tglobal.showCriteria()
*----  criteria -----*
ec01 'Economical aspect/criterion of objective Eco'
  Scale = (0.0, 100.0)
  Weight = 0.111
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12219873150105708
  Threshold pref : 10.00 + 0.00x ; percentile:  0.2406143891344737
  Threshold veto : 60.00 + 0.00x ; percentile:  0.9518621735323638

so02 'Societal aspect/criterion of objective Soc'
  Scale = (0.0, 100.0)
  Weight = 0.067
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12391856904414567
  Threshold pref : 10.00 + 0.00x ; percentile:  0.2441835219282682

en03 'Environmental aspect/criterion of objective Env'
  Scale = (0.0, 100.0)
  Weight = 0.048
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12122229880337092
  Threshold pref : 10.00 + 0.00x ; percentile:  0.23899260947699674

so04 'Societal aspect/criterion of objective Soc'
  Scale = (0.0, 100.0)
  Weight = 0.067
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12268731709130695
  Threshold pref : 10.00 + 0.00x ; percentile:  0.2425739734540066

so05 'Societal aspect/criterion of objective Soc'
  Scale = (0.0, 100.0)
  Weight = 0.067
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12367490977427552
  Threshold pref : 10.00 + 0.00x ; percentile:  0.2440878125867555

so06 'Societal aspect/criterion of objective Soc'
  Scale = (0.0, 100.0)
  Weight = 0.067
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12250882125805859
  Threshold pref : 10.00 + 0.00x ; percentile:  0.24220219987456107

ec07 'Economical aspect/criterion of objective Eco'
  Scale = (0.0, 100.0)
  Weight = 0.111
  Threshold ind : 5.00 + 0.00x ; percentile:  0.12204602504961296
```

In all the project we suppose this convention + stands for Good, - stands for Weak and ~ for Fair

According to the documentation we always need at least 3 decision actions, so here,the case being satisfied, we can work with the data set. Its 15 criteria are distributed across 3 equisignificant objectives: the economy, the environment and society. Each of these criteria is attributed a varying weight and must be maximized during the process of selecting the best policy.

t.showActions() : Tail of the output

```
key:  p1992
  short name: p1992
  name:       action p1992 Eco- Soc~ Env+
  comment:    random public polivy

key:  p1993
  short name: p1993
  name:       action p1993 Eco- Soc~ Env+
  comment:    random public polivy

key:  p1994
  short name: p1994
  name:       action p1994 Eco- Soc~ Env-
  comment:    random public polivy

key:  p1995
  short name: p1995
  name:       action p1995 Eco- Soc~ Env-
  comment:    random public polivy

key:  p1996
  short name: p1996
  name:       action p1996 Eco+ Soc~ Env-
  comment:    random public polivy

key:  p1997
  short name: p1997
  name:       action p1997 Eco- Soc- Env+
  comment:    random public polivy

key:  p1998
  short name: p1998
  name:       action p1998 Eco+ Soc~ Env+
  comment:    random public polivy

key:  p1999
  short name: p1999
  name:       action p1999 Eco- Soc+ Env~
  comment:    random public polivy

key:  p2000
  short name: p2000
  name:       action p2000 Eco+ Soc- Env+
  comment:    random public polivy
```

t.exportGraphViz('testgraph_23')

After running the previous code in python 3, i can now see the graph. But unfortunately, it is very big graph and it is unclear to know any thing from it.(See Figure 1)

As i can observe that the graph is unclear and big, but by making close observation we can make out that there are alternatives that are dominants on others.

t.showPerformanceTableau()

(based on xml : The old ancestor of show.HTMLheatmap Function)

And with t.showStatistics we can go much deeper to look in detail in our data: we have 2000 actions, 15 critera and 3 respective objectives.

```
*-------- Performance tableau summary statistics -------*
Instance name      : perTab_23
#Actions           : 2000
#Criteria          : 15
*Statistics per Criterion*
Criterion name       : ec01
Criterion weight     : 35
  criterion scale      : 0.00 - 100.00
  # missing evaluations : 64
  mean evaluation       : 50.45
  standard deviation    : 21.92
  maximal evaluation    : 98.80
  quantile Q3 (x_75)    : 67.06
  median evaluation     : 50.82
  quantile Q1 (x_25)    : 33.11
  minimal evaluation    : 1.70
  mean absolute difference      : 25.21
  standard difference deviation : 31.00
Criterion name       : ec07
Criterion weight     : 35
  criterion scale      : 0.00 - 100.00
  # missing evaluations : 49
  mean evaluation       : 50.67
  standard deviation    : 21.85
  maximal evaluation    : 97.41
  quantile Q3 (x_75)    : 67.88
  median evaluation     : 50.30
  quantile Q1 (x_25)    : 33.90
  minimal evaluation    : 2.49
  mean absolute difference      : 25.14
```

(…………..…..)

2 Construct the corresponding bipolar-valued outranking digraph and enumerate its chordless circuits. How are such circuits, the case given, influencing the algorithmic decision aid tools?

In this Digraph3 module, the main outrankingDigraphs.BipolarOutrankingDigraph class provides a generic bipolar-valued outranking digraph model. A given object of this class consists in

1. a potential set of decision actions : a dictionary describing the potential decision actions or alternatives with 'name' and 'comment' attributes,

2. a coherent family of criteria: a dictionary of criteria functions used for measuring the performance of each potential decision action with respect to the preference dimension captured by each criterion,

3. the evaluations: a dictionary of performance evaluations for each decision action or alternative on each criterion function.

4. the digraph valuationdomain, a dictionary with three entries: the minimum (- 100, means certainly no link), the median (0, means missing information) and the maximum characteristic value (+100, means certainly a link),

5. the outranking relation : a double dictionary defined on the Cartesian product of the set of decision alternatives capturing the credibility of the pairwise outranking situation computed on the basis of the performance differences observed between couples of decision alternatives on the given family if criteria functions.

Initially, I started with building the bipolar-valued outranking digraph by using this command: *globalPoint* = *BipolarOutrankingDigraph*(*tl*) . To find the best alternatives, i have to search for the the alternatives that have the benefit criteria is maximized and also have the cost criteria is minimized. The weight assigned to each criteria illustrates how important the criteria is in determining the best alternatives of the process.

According to the definition presented in the lecture notes : we say that a decision alternative x outranks a decision alternative y if :

• A potentially weighted majority of criteria validates the statement x performance at least as good as y and,

• No considerably large negative performance difference is observed in disfavour of x. The table below (Figure 1) shows us the Rubis Performance Table along with inputs from the Rubis family of criteria.

Because of there are some criteria have another attributes like the weight, the scale and the thresholds, i putted it also in the end of the table, to see a briefly view about whole problem (the alternative and the family of criteria). Let's see briefly also the thresholds attributes.

The threshold attributes in some criteria give us the limit up to which a difference of performance of a given alternative is considered the same(indifference), considered better(preference) or considered unacceptable compared to another alternative (veto). When we talk of a veto we deal with two distinct principles:

a) No considerable **negative** performance difference is observed between 'a' and 'b' on any criterion.

b) No considerable **positive** performance difference is observed between 'a' and 'b' on any criterion.

The concept of Veto threshold allows us to model the fact that the performance difference observed between two potential decision alternatives on a criterion may be:

a) Either attesting the presence of counter-performance large enough to put to doubt a significantly affirmed outranking situation.
b) Or, attesting the presence of an out-performance large enough to put to doubt a significantly refuted outranking solution. We take into account discrimination threshold on each of the criterion as we deal with imprecision, uncertainty and difficulty in quantification.

criteria	en09	ec13	ec07	en15	ec01	en14	so06	en03	en11	so04	en12	en10	so02	so08	so05
weights	+15.00	+35.00	+35.00	+15.00	+35.00	+15.00	+21.00	+15.00	+15.00	+21.00	+15.00	+15.00	+21.00	+21.00	+21.00
tau(*)	+0.38	+0.33	+0.33	+0.31	+0.30	+0.29	+0.25	+0.25	+0.25	+0.25	+0.23	+0.21	+0.19	+0.17	+0.14
p016	76	79	83	55	73	93	74	69	70	63	36	70	51	58	NA
p081	70	69	59	83	79	58	57	90	68	82	56	93	77	68	54
p017	64	81	75	85	45	79	71	59	36	32	83	79	55	70	92
p014	47	53	81	87	62	83	50	62	81	64	79	80	60	46	50
p099	71	66	88	87	56	80	50	57	72	71	41	55	45	29	91
p051	64	56	68	81	44	96	43	92	80	53	91	84	28	44	43
p090	91	51	55	51	19	90	84	74	90	79	49	79	72	30	79
p053	60	86	91	61	65	66	46	57	63	48	88	83	61	51	38
p067	59	84	82	55	83	72	44	63	22	59	44	26	77	63	34
p094	75	72	65	63	69	71	59	54	56	62	75	NA	20	54	69
p013	73	95	95	24	79	23	45	44	31	79	NA	41	71	73	72
p068	90	80	52	77	66	89	36	94	85	15	94	48	72	15	24
p006	62	40	88	65	46	85	51	89	76	48	NA	52	49	64	NA
p005	71	87	85	70	54	61	63	91	45	75	61	76	18	50	15
p074	56	61	93	73	66	69	17	90	87	NA	69	75	39	42	31
p087	67	56	61	78	38	69	45	81	51	56	73	69	46	47	96
p091	73	51	37	92	45	70	73	83	78	42	70	NA	34	62	29
p093	57	84	37	37	63	42	81	24	25	83	25	33	55	93	37
p066	89	NA	69	41	72	17	64	36	62	66	51	70	42	63	43
p012	74	59	75	95	61	34	29	68	57	49	71	62	22	47	12
p098	49	72	77	73	46	54	41	77	91	NA	83	76	24	50	37
p085	85	32	33	41	40	53	61	84	72	56	86	82	71	67	41
p040	48	40	42	34	64	74	74	18	54	91	NA	25	64	69	76
p056	58	58	77	49	65	36	50	60	44	42	47	35	64	62	68

11.11%	22.22%	33.33%	44.44%	55.55%	66.66%	77.77%	88.88%	100.00%

(*) tau: *Ordinal (Kendall) correlation between marginal criterion and global ranking relation*
Ranking rule: **NetFlows**
Ordinal (Kendall) correlation between global ranking and global outranking relation: **+0.839**
Mean marginal correlation (a) : **+0.266**
Standard marginal correlation deviation (b) : **+0.065**
Ranking fairness (a) - (b) : **+0.201**

Figure 1: Performance Heatmap from the Global Point of View

It is worthwhile noticing that green and red marked evaluations indicate best, respectively worst, performances of an alternative on a criterion. In this example, we may hence notice that alternative p016 is in fact best performing highest quitile in 9 (as choosen in the script) out of 15 criteria.

Note: Missing (NA) evaluation are registered in a performance tableau as Decimal('- 999') value

```
gpChordlessCircuits=BipolarOutrankingDigraph(tl)
gpChordlessCircuits.exportGraphViz('globalPoint')
gpChordlessCircuits.computeChordlessCircuits()
gpChordlessCircuits.showChordlessCircuits()
```

Listing 2: Python Code for generating Chordless Circuits Python Commands

Results: # Here we can show 31 circuits out of 110.

```
*---- Chordless circuits ----*
110 circuits.
1:  ['p001', 'p071', 'p040'] , credibility : 0.095
2:  ['p001', 'p071', 'p010'] , credibility : 0.159
3:  ['p001', 'p082', 'p040'] , credibility : 0.117
4:  ['p002', 'p078', 'p012'] , credibility : 0.213
5:  ['p002', 'p078', 'p015'] , credibility : 0.549
6:  ['p002', 'p078', 'p080'] , credibility : 0.241
7:  ['p002', 'p078', 'p010'] , credibility : 0.184
8:  ['p002', 'p078', 'p052'] , credibility : 0.190
9:  ['p002', 'p078', 'p082'] , credibility : 0.254
10:  ['p002', 'p043', 'p015'] , credibility : 0.149
11:  ['p002', 'p043', 'p019', 'p080'] , credibility : 0.149
12:  ['p002', 'p043', 'p082'] , credibility : 0.149
13:  ['p002', 'p088', 'p015'] , credibility : 0.238
14:  ['p006', 'p033', 'p093'] , credibility : 0.140
15:  ['p006', 'p098', 'p090'] , credibility : 0.117
16:  ['p008', 'p088', 'p069'] , credibility : 0.156
17:  ['p008', 'p088', 'p020'] , credibility : 0.032
18:  ['p009', 'p015', 'p029'] , credibility : 0.238
19:  ['p009', 'p031', 'p018'] , credibility : 0.343
20:  ['p010', 'p080', 'p039'] , credibility : 0.003
21:  ['p010', 'p100', 'p023'] , credibility : 0.067
22:  ['p010', 'p058', 'p039'] , credibility : 0.114
23:  ['p010', 'p058', 'p078'] , credibility : 0.114
24:  ['p010', 'p058', 'p023'] , credibility : 0.067
25:  ['p010', 'p030', 'p023'] , credibility : 0.067
26:  ['p010', 'p070', 'p023'] , credibility : 0.067
27:  ['p012', 'p040', 'p075'] , credibility : 0.076
28:  ['p012', 'p040', 'p078'] , credibility : 0.419
29:  ['p012', 'p041', 'p078'] , credibility : 0.419
30:  ['p012', 'p042', 'p078'] , credibility : 0.419
31:  ['p012', 'p058', 'p078'] , credibility : 0.419
```

3 What are apparently the 5 best-ranked decision alternatives in your decision problem from the different decision objectives point of views and from a global fair compromise view? Justify your ranking approach from a methodological point of view.

- To evaluate the five best-ranked decision alternatives, from the three different point of views, I decided to first show a brief representation of their Heatmaps. (For the global Point of view, the Heatmap is given in question 1.

The Heatmaps for specifically the Economical, Social and Environmental decision objectives are also represented in the respective figures below.

Results based on the Heatmaps

criteria	ec01	ec13	ec07
weights	+35.00	+35.00	+35.00
tau(*)	+0.58	+0.56	+0.52
p013	79	95	95
p067	83	84	82
p053	65	86	91
p016	73	79	83
p071	91	80	62
p018	84	76	67
p041	84	61	76
p019	87	61	75
p078	61	71	95
p005	54	87	85
p074	66	61	93
p073	78	71	61
p015	83	56	71
p081	79	69	59
p094	69	72	65
p099	56	66	88
p033	67	89	50
p056	65	58	77
p068	66	80	52
p069	43	77	89
p014	62	53	81
p012	61	59	75
p017	45	81	75
p082	89	50	55
p066	72	NA	69
p098	46	72	77
p043	65	74	NA

Performance Heatmap from the Economical Point of View

criteria	so08	so06	so04	so02	so05
weights	+21.00	+21.00	+21.00	+21.00	+21.00
tau(*)	+0.58	+0.55	+0.51	+0.51	+0.45
p040	69	74	91	64	76
p010	62	76	82	78	74
p028	74	67	68	66	84
p061	79	81	53	80	72
p022	76	72	79	73	59
p058	88	93	55	72	61
p008	77	65	62	84	72
p093	93	81	83	55	37
p081	68	57	82	77	54
p013	73	45	79	71	72
p052	51	96	80	52	72
p090	30	84	79	72	79
p036	73	80	71	61	40
p029	73	66	52	48	91
p047	75	74	51	55	64
p042	73	43	90	54	72
p026	87	50	68	43	74
p062	74	46	55	85	64
p017	70	71	32	55	92
p041	70	42	71	75	56
p055	63	53	42	94	74
p096	NA	74	55	52	78
p085	67	61	56	71	41
p016	58	74	63	51	NA
p039	67	55	52	NA	72
p021	51	76	54	64	43
p002	69	63	58	42	NA

PH from the Social Point of View

criteria	en03	en14	en10	en11	en09	en15	en12
weights	+15.00	+15.00	+15.00	+15.00	+15.00	+15.00	+15.00
tau(*)	+0.56	+0.54	+0.54	+0.53	+0.51	+0.51	+0.40
p051	92	96	84	80	64	81	91
p068	94	89	48	85	90	77	94
p095	76	72	99	54	71	85	89
p091	83	70	NA	78	73	92	70
p031	95	73	72	67	87	67	64
p090	74	90	79	90	91	51	49
p074	90	69	75	87	56	73	69
p014	62	83	80	81	47	87	79
p081	90	58	93	68	70	83	56
p004	88	62	76	77	75	66	65
p097	64	90	63	71	74	84	62
p011	72	71	83	89	58	86	49
p085	84	53	82	72	85	41	86
p080	90	77	57	86	67	46	77
p098	77	54	76	91	49	73	83
p006	89	85	52	76	62	65	NA
p087	81	69	69	51	67	78	73
p076	71	55	86	45	68	93	72
p017	59	79	79	36	64	85	83
p061	79	52	80	50	68	80	75
p016	69	93	70	70	76	55	36
p053	57	66	83	63	60	61	88
p099	57	80	55	72	71	87	41
p005	91	61	76	45	71	70	61
p030	75	69	80	82	63	64	34
p065	75	56	92	57	72	42	82
p075	64	58	79	88	86	55	30

PH from the Environmental Point of View

➔ Top 5 'globally' {p016 p081 p017 p014 p099 }

- Best-5 from the economical point of view {p013, p067, p053, p016, p071}
- Best-5 from the social point of view {p040, p010, p028, p061, p022}
- Best-5 from the environemental point of view {p051, p068, p095, p091, p031}

Studying the perspective: performance.

Applying ranking: CopelandOrder

['p081', 'p016', 'p099', 'p013', 'p017', 'p005', 'p014', 'p053', 'p094', 'p067', 'p090', 'p068', 'p074', 'p051', 'p006', 'p098', 'p066', 'p087', 'p093', 'p056', 'p085', 'p037', 'p075', 'p091', 'p021', 'p040', 'p012', 'p045', 'p028', 'p061', 'p077', 'p036', 'p041', 'p033', 'p047', 'p096', 'p073', 'p018', 'p076', 'p010', 'p039', 'p062', 'p078', 'p097', 'p071', 'p069', 'p052', 'p058', 'p019', 'p022', 'p031', 'p009', 'p034', 'p082', 'p100', 'p070', 'p042', 'p030', 'p072', 'p043', 'p001', 'p023', 'p083', 'p002', 'p032', 'p065', 'p048', 'p029', 'p015', 'p080', 'p050', 'p026', 'p088', 'p095', 'p038', 'p025', 'p084', 'p092', 'p049', 'p011', 'p055', 'p024', 'p008', 'p046', 'p020', 'p086', 'p089', 'p044', 'p004', 'p054', 'p063', 'p035', 'p064', 'p079', 'p060', 'p027', 'p003', 'p057', 'p059', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.840

Epistemic determination : 0.387

Bipolar-valued equivalence : +0.325

CopelandOrder Score: 0.8399911741235921316980024045

Applying ranking: NetFlowsOrder

['p016', 'p081', 'p017', 'p014', 'p099', 'p051', 'p090', 'p053', 'p067', 'p094', 'p013', 'p068', 'p006', 'p005', 'p074', 'p087', 'p091', 'p093', 'p066', 'p012', 'p098', 'p085', 'p040', 'p056', 'p045', 'p061', 'p037', 'p075', 'p021', 'p036', 'p071', 'p028', 'p097', 'p076', 'p033', 'p041', 'p077', 'p047', 'p039', 'p031', 'p073', 'p096', 'p018', 'p082', 'p078', 'p019', 'p022', 'p062', 'p052', 'p069', 'p030', 'p058', 'p034', 'p010', 'p100', 'p009', 'p001', 'p070', 'p072', 'p002', 'p032', 'p080', 'p042', 'p023', 'p095', 'p048', 'p083', 'p065', 'p015', 'p029', 'p043', 'p050', 'p026', 'p092', 'p025', 'p011', 'p038', 'p084', 'p049', 'p088', 'p046', 'p008', 'p004', 'p055', 'p063', 'p020', 'p089', 'p086', 'p054', 'p024', 'p035', 'p064', 'p044', 'p060', 'p003', 'p079', 'p027', 'p059', 'p007', 'p057']

Correlation indexes:

Crisp ordinal correlation : +0.839

Epistemic determination : 0.387

Bipolar-valued equivalence : +0.324

NetFlowsOrder Score: 0.8389758665368223195133159643

Applying ranking: KohlerOrder

['p081', 'p005', 'p068', 'p017', 'p016', 'p099', 'p051', 'p014', 'p013', 'p067', 'p074', 'p053', 'p094', 'p098', 'p090', 'p085', 'p061', 'p056', 'p066', 'p087', 'p006', 'p028', 'p093', 'p096', 'p097', 'p037', 'p012', 'p058', 'p041', 'p021', 'p036', 'p076', 'p062', 'p091', 'p039', 'p033', 'p069', 'p075', 'p078', 'p019', 'p073', 'p009', 'p032', 'p100', 'p083', 'p077', 'p052', 'p071', 'p047', 'p010', 'p022', 'p045', 'p080', 'p048', 'p042', 'p082', 'p040', 'p070', 'p034', 'p072', 'p031', 'p018', 'p001', 'p065', 'p030', 'p049', 'p023', 'p050', 'p002', 'p043', 'p038', 'p025', 'p026', 'p011', 'p084', 'p020', 'p092', 'p008', 'p046', 'p054', 'p088', 'p089', 'p095', 'p015', 'p024', 'p055', 'p035', 'p029', 'p027', 'p064', 'p086', 'p003', 'p044', 'p004', 'p063', 'p079', 'p060', 'p059', 'p007', 'p057']

Correlation indexes:

Crisp ordinal correlation : +0.819

Epistemic determination : 0.387

Bipolar-valued equivalence : +0.317

KohlerOrder Score: 0.8194029925029820512863051356

['p005', 'p081']

['p005', 'p013', 'p016', 'p017', 'p081']

Studying the perspective: economy.

Applying ranking: CopelandOrder

['p013', 'p067', 'p053', 'p016', 'p071', 'p018', 'p005', 'p078', 'p041', 'p019', 'p073', 'p074', 'p069', 'p033', 'p066', 'p081', 'p015', 'p094', 'p043', 'p099', 'p096', 'p068', 'p017', 'p056', 'p014', 'p093', 'p098', 'p012', 'p062', 'p082', 'p037', 'p077', 'p035', 'p010', 'p084', 'p047', 'p075', 'p023', 'p097', 'p020', 'p032', 'p045', 'p083', 'p051', 'p100', 'p006', 'p009', 'p087', 'p042', 'p060', 'p040', 'p076', 'p036', 'p028', 'p048', 'p001', 'p089', 'p092', 'p021', 'p091', 'p065', 'p090', 'p044', 'p027', 'p080', 'p025', 'p029', 'p052', 'p034', 'p024', 'p064', 'p022', 'p085', 'p050', 'p063', 'p072', 'p079', 'p026', 'p038', 'p058', 'p002', 'p055', 'p057', 'p070', 'p086', 'p095', 'p031', 'p011', 'p008', 'p049', 'p003', 'p039', 'p054', 'p088', 'p046', 'p061', 'p059', 'p004', 'p030', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.898

Epistemic determination : 0.625

Bipolar-valued equivalence : +0.562

CopelandOrder Score: 0.8982558139534883720930233061

Applying ranking: NetFlowsOrder

['p013', 'p067', 'p053', 'p016', 'p071', 'p018', 'p041', 'p019', 'p078', 'p005', 'p074', 'p073', 'p015', 'p081', 'p094', 'p099', 'p033', 'p056', 'p068', 'p069', 'p014', 'p012', 'p017', 'p082', 'p066', 'p098', 'p043', 'p096', 'p037', 'p093', 'p010', 'p035', 'p062', 'p084', 'p047', 'p077', 'p045', 'p075', 'p097', 'p032', 'p020', 'p051', 'p006', 'p083', 'p023', 'p009', 'p087', 'p042', 'p040', 'p036', 'p100', 'p076', 'p001', 'p048', 'p060', 'p028', 'p092', 'p091', 'p089', 'p021', 'p044', 'p065', 'p090', 'p025', 'p027', 'p080', 'p029', 'p063', 'p034', 'p064', 'p052', 'p038', 'p024', 'p072', 'p002', 'p085', 'p079', 'p022', 'p050', 'p026', 'p039', 'p088', 'p008', 'p055', 'p057', 'p086', 'p070', 'p058', 'p095', 'p011', 'p031', 'p049', 'p003', 'p061', 'p054', 'p046', 'p030', 'p059', 'p004', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.889

Epistemic determination : 0.625

Bipolar-valued equivalence : +0.556

NetFlowsOrder Score: 0.8892118863049095607235142648

Applying ranking: KohlerOrder

['p013', 'p067', 'p053', 'p016', 'p078', 'p071', 'p043', 'p005', 'p018', 'p074', 'p062', 'p099', 'p098', 'p096', 'p017', 'p081', 'p073', 'p014', 'p094', 'p093', 'p087', 'p084', 'p082', 'p077', 'p075', 'p069', 'p056', 'p041', 'p019', 'p015', 'p066', 'p068', 'p033', 'p012', 'p097', 'p037', 'p083', 'p047', 'p035', 'p051', 'p045', 'p032', 'p023', 'p010', 'p020', 'p009', 'p042', 'p100', 'p092', 'p089', 'p036', 'p006', 'p091', 'p090', 'p048', 'p040', 'p060', 'p085', 'p080', 'p079', 'p076', 'p028', 'p044', 'p021', 'p001', 'p065', 'p064', 'p050', 'p034', 'p022', 'p095', 'p088', 'p086', 'p072', 'p070', 'p063', 'p025', 'p059', 'p058', 'p057', 'p055', 'p052', 'p038', 'p029', 'p027', 'p024', 'p026', 'p054', 'p031', 'p011', 'p061', 'p049', 'p008', 'p030', 'p004', 'p003', 'p002', 'p046', 'p039', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.854

Epistemic determination : 0.625

Bipolar-valued equivalence : +0.534

KohlerOrder Score: 0.8541128337639965546942291684

['p013']

['p013', 'p053']

Studying the perspective: environment.

Applying ranking: CopelandOrder

['p068', 'p051', 'p095', 'p091', 'p090', 'p081', 'p031', 'p074', 'p014', 'p006', 'p004', 'p085', 'p011', 'p080', 'p097', 'p098', 'p087', 'p017', 'p061', 'p030', 'p005', 'p076', 'p065', 'p099', 'p075', 'p053', 'p016', 'p038', 'p045', 'p094', 'p012', 'p039', 'p046', 'p021', 'p086', 'p070', 'p025', 'p078', 'p071', 'p088', 'p100', 'p049', 'p023', 'p048', 'p019', 'p037', 'p063', 'p072', 'p034', 'p002', 'p001', 'p092', 'p082', 'p032', 'p066', 'p067', 'p050', 'p058', 'p036', 'p083', 'p073', 'p009', 'p052', 'p056', 'p015', 'p089', 'p028', 'p033', 'p077', 'p040', 'p022', 'p013', 'p054', 'p069', 'p047', 'p018', 'p020', 'p093', 'p084', 'p060', 'p059', 'p096', 'p035', 'p062', 'p024', 'p057', 'p043', 'p064', 'p027', 'p055', 'p029', 'p003', 'p042', 'p079', 'p041', 'p026', 'p008', 'p044', 'p010', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.915

Epistemic determination : 0.584

Bipolar-valued equivalence : +0.534

CopelandOrder Score: 0.9151608038958791684176699456

Applying ranking: NetFlowsOrder

['p051', 'p068', 'p095', 'p091', 'p031', 'p090', 'p074', 'p014', 'p081', 'p004', 'p097', 'p011', 'p085', 'p080', 'p098', 'p006', 'p087', 'p076', 'p017', 'p061', 'p016', 'p053', 'p099', 'p005', 'p030', 'p065', 'p075', 'p045', 'p012', 'p038', 'p094', 'p039', 'p046', 'p021', 'p070', 'p086', 'p025', 'p071', 'p078', 'p088', 'p100', 'p049', 'p048', 'p019', 'p063', 'p092', 'p023', 'p072', 'p066', 'p037', 'p034', 'p002', 'p032', 'p001', 'p067', 'p082', 'p036', 'p050', 'p009', 'p058', 'p052', 'p073', 'p083', 'p056', 'p040', 'p015', 'p089', 'p028', 'p077', 'p033', 'p022', 'p013', 'p054', 'p047', 'p069', 'p084', 'p020', 'p018', 'p093', 'p059', 'p043', 'p029', 'p096', 'p060', 'p035', 'p062', 'p064', 'p024', 'p055', 'p057', 'p079', 'p003', 'p042', 'p026', 'p027', 'p044', 'p010', 'p008', 'p041', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.915

Epistemic determination : 0.584

Bipolar-valued equivalence : +0.534

NetFlowsOrder Score: 0.9147158430771512619583220596

Applying ranking: KohlerOrder

['p068', 'p051', 'p095', 'p014', 'p031', 'p004', 'p091', 'p081', 'p090', 'p080', 'p061', 'p011', 'p098', 'p085', 'p087', 'p005', 'p097', 'p076', 'p074', 'p016', 'p075', 'p006', 'p065', 'p017', 'p053', 'p045', 'p099', 'p038', 'p046', 'p094', 'p030', 'p012', 'p086', 'p070', 'p039', 'p072', 'p071', 'p025', 'p021', 'p049', 'p078', 'p088', 'p100', 'p023', 'p019', 'p092', 'p083', 'p048', 'p036', 'p034', 'p032', 'p063', 'p037', 'p066', 'p002', 'p077', 'p067', 'p001', 'p082', 'p073', 'p089', 'p058', 'p056', 'p052', 'p050', 'p028', 'p009', 'p054', 'p033', 'p047', 'p015', 'p022', 'p013', 'p093', 'p069', 'p040', 'p060', 'p035', 'p020', 'p096', 'p059', 'p057', 'p027', 'p024', 'p018', 'p084', 'p064', 'p043', 'p042', 'p029', 'p062', 'p055', 'p008', 'p079', 'p041', 'p003', 'p044', 'p010', 'p026', 'p007']

Correlation indexes:

Crisp ordinal correlation : +0.909

Epistemic determination : 0.584

Bipolar-valued equivalence : +0.530

KohlerOrder Score: 0.9087335920698094084493115975

['p051', 'p068']

['p006', 'p051', 'p068', 'p090']

Studying the perspective: society.

Applying ranking: CopelandOrder

['p090', 'p058', 'p093', 'p010', 'p040', 'p061', 'p022', 'p028', 'p008', 'p013', 'p052', 'p081', 'p036', 'p042', 'p017', 'p026', 'p029', 'p047', 'p096', 'p041', 'p062', 'p055', 'p016', 'p039', 'p002', 'p085', 'p034', 'p030', 'p021', 'p066', 'p056', 'p077', 'p007', 'p099', 'p067', 'p070', 'p024', 'p069', 'p006', 'p094', 'p014', 'p037', 'p087', 'p050', 'p072', 'p054', 'p018', 'p091', 'p033', 'p053', 'p031', 'p049', 'p009', 'p043', 'p005', 'p003', 'p001', 'p100', 'p051', 'p044', 'p048', 'p088', 'p098', 'p045', 'p084', 'p073', 'p083', 'p064', 'p079', 'p075', 'p076', 'p032', 'p074', 'p023', 'p020', 'p078', 'p025', 'p012', 'p095', 'p059', 'p092', 'p080', 'p011', 'p027', 'p097', 'p038', 'p065', 'p068', 'p082', 'p086', 'p004', 'p060', 'p046', 'p019', 'p057', 'p035', 'p063', 'p015', 'p089', 'p071']

Correlation indexes:

Crisp ordinal correlation : +0.903

Epistemic determination : 0.581

Bipolar-valued equivalence : +0.524

CopelandOrder Score: 0.9033234731164085609883417435

Applying ranking: NetFlowsOrder

['p040', 'p010', 'p028', 'p061', 'p022', 'p058', 'p008', 'p093', 'p081', 'p013', 'p052', 'p090', 'p036', 'p029', 'p047', 'p042', 'p026', 'p062', 'p017', 'p041', 'p055', 'p096', 'p085', 'p016', 'p039', 'p021', 'p002', 'p034', 'p066', 'p056', 'p077', 'p007', 'p030', 'p067', 'p024', 'p099', 'p014', 'p087', 'p072', 'p070', 'p094', 'p006', 'p069', 'p037', 'p050', 'p054', 'p033', 'p018', 'p053', 'p091', 'p043', 'p049', 'p001', 'p031', 'p009', 'p005', 'p003', 'p100', 'p051', 'p044', 'p064', 'p098', 'p084', 'p045', 'p083', 'p048', 'p088', 'p075', 'p076', 'p032', 'p073', 'p074', 'p079', 'p023', 'p025', 'p068', 'p012', 'p082', 'p092', 'p078', 'p020', 'p059', 'p095', 'p038', 'p011', 'p004', 'p097', 'p065', 'p080', 'p086', 'p027', 'p060', 'p046', 'p019', 'p089', 'p035', 'p057', 'p063', 'p071', 'p015']

Correlation indexes:

Crisp ordinal correlation : +0.900

Epistemic determination : 0.581

Bipolar-valued equivalence : +0.522

NetFlowsOrder Score: 0.8999825996171915782147207239

Applying ranking: KohlerOrder

['p061', 'p022', 'p013', 'p010', 'p090', 'p081', 'p058', 'p040', 'p093', 'p028', 'p052', 'p008', 'p042', 'p036', 'p029', 'p017', 'p026', 'p047', 'p096', 'p062', 'p094', 'p067', 'p055', 'p041', 'p066', 'p016', 'p077', 'p039', 'p030', 'p085', 'p021', 'p034', 'p007', 'p002', 'p056', 'p014', 'p070', 'p072', 'p024', 'p054', 'p018', 'p006', 'p100', 'p099', 'p050', 'p037', 'p053', 'p003', 'p091', 'p087', 'p009', 'p088', 'p084', 'p083', 'p073', 'p069', 'p031', 'p051', 'p049', 'p044', 'p005', 'p033', 'p043', 'p079', 'p075', 'p001', 'p048', 'p045', 'p098', 'p064', 'p092', 'p080', 'p076', 'p086', 'p078', 'p074', 'p032', 'p095', 'p059', 'p020', 'p025', 'p023', 'p011', 'p012', 'p097', 'p065', 'p027', 'p082', 'p060', 'p038', 'p035', 'p089', 'p057', 'p046', 'p015', 'p004', 'p068', 'p019', 'p063', 'p071']

Correlation indexes:

Crisp ordinal correlation : +0.882

Epistemic determination : 0.581

Bipolar-valued equivalence : +0.512

KohlerOrder Score: 0.8818862014964329215242735340

['p010', 'p013', 'p022', 'p061']

['p010', 'p013', 'p022', 'p040', 'p058', 'p061', 'p090']

4 How would you rate your 100 public policies into relative deciles classes ?

QuantilesSortingDigraph coupled with the appropriate number of bins, we can easily order our public policies into deciles of the relative sort. The procedure is straightforward and we report the whole output of the deciles showing p013 to be at the top of our rating.

]0.90-1.00] : ['p013']

]0.80-0.90] : ['p016', 'p017', 'p068', 'p081']

]0.70-0.80] : ['p005', 'p010', 'p014', 'p040', 'p053', 'p061', 'p066', 'p067', 'p074', 'p090', 'p094', 'p099']

]0.60-0.70] : ['p006', 'p012', 'p018', 'p021', 'p028', 'p037', 'p039', 'p041', 'p043', 'p045', 'p047', 'p051', 'p056', 'p058', 'p062', 'p071', 'p077', 'p085', 'p087', 'p093', 'p096', 'p097', 'p098', 'p100']

]0.50-0.70] : ['p031', 'p078']

]0.50-0.60] : ['p019', 'p025', 'p029', 'p032', 'p033', 'p034', 'p036', 'p052', 'p070', 'p072', 'p073', 'p075', 'p076', 'p082', 'p091']

]0.40-0.50] : ['p001', 'p002', 'p009', 'p015', 'p022', 'p023', 'p024', 'p026', 'p030', 'p038', 'p042', 'p048', 'p049', 'p050', 'p065', 'p069', 'p080', 'p083', 'p084', 'p088', 'p089', 'p092']

]0.30-0.40] : ['p011', 'p020', 'p044', 'p054', 'p055', 'p086', 'p095']

]0.20-0.40] : ['p046']

]0.20-0.30] : ['p003', 'p008', 'p027', 'p035', 'p059', 'p060', 'p063', 'p064', 'p079']

]0.10-0.30] : ['p004']

]0.10-0.20] : ['p007', 'p057']

5 Using the given historical records in ***historicalData$_x$.py***, how would you rate your 100 public policies into ab- solute deciles classes ? Explain the differences you may observe between the absolute and the previous relative rating result.

Answer:

To rate the 100 public policies into absolute decile classes, I implemented the incremental performance quantiles representation of the given historical records.The Python Code I followed is shown in the last appendice.

- We know that during the rating, the classifier will choose between absolute and relative rating. While the *relative rating*, takes solely itself as a reference and lacks generalization, it is often the only option when considering a unique dataset. In contrast, the *absolute rating* requires additional data (can be some data augmentation). It is important to favor absolute rating when possible or at least give him a large variety of more weights like in a statistical method called Holwinters that give more importance to new data to make a global interpretation but without neglecting the impact of old data.

[0.70 - 0.80[['p081', 'p016', 'p017', 'p014', 'p099', 'p053']

[0.60 - 0.70[['p090', 'p051', 'p067', 'p094', 'p013', 'p068', 'p006', 'p005', 'p074', 'p087', 'p091', 'p066', 'p093', 'p098', 'p012', 'p085', 'p040']

[0.50 - 0.60[['p056', 'p045', 'p061', 'p037', 'p075', 'p021', 'p036', 'p071', 'p028', 'p097', 'p076', 'p033', 'p041', 'p077', 'p047', 'p039', 'p073', 'p096', 'p031', 'p018', 'p078', 'p082', 'p019', 'p022', 'p062']

[0.40 - 0.50[['p052', 'p069', 'p030', 'p010', 'p058', 'p034', 'p100', 'p009', 'p001', 'p070', 'p072', 'p002', 'p032', 'p080', 'p023', 'p042', 'p095', 'p048', 'p083', 'p065', 'p015', 'p029', 'p043', 'p050', 'p026', 'p092', 'p025']

[0.30 - 0.40[['p038', 'p011', 'p084', 'p049', 'p088', 'p046', 'p008', 'p004', 'p055', 'p063', 'p020', 'p086', 'p089']

[0.20 - 0.30[['p024', 'p054', 'p035', 'p064', 'p044', 'p060', 'p003', 'p079', 'p027', 'p059']

[0.10 - 0.20[['p007', 'p057']

6 Select among your 100 potential policies a shortlist of up to 15 potential best policies, all reaching an abso- lute performance quantile of at least 66.67%.

For sorting we use sortingDigraph Class, which specialize on sorting of a large set of alternatives into quantiles delimited ordered classes.

NOTE: We notice that we found just 12 elements that are reaching the said performance value of at least 66.67%

[(0.8446601941747572, 'p013'),
(0.8110957004160887, 'p016'),
(0.7875635691169671, 'p017'),
(0.7673139158576052, 'p081'),
(0.7337031900138696, 'p053'),
(0.7295423023578363, 'p005'),
(0.7267683772538142, 'p067'),
(0.7152103559870551, 'p094'),
(0.7081830790568655, 'p066'),
(0.7067961165048543, 'p068'),
(0.6965325936199723, 'p014'),
(0.6880258899676376, 'p074'), then ….

(0.663384188626907l, 'm3'), (0.6599167822468793, 'p099'), (0.6339805825242718, 'p061'), (0.6185852981969487, 'p077'),
(0.6178918169209431, 'p012'), (0.6098474341192788, 'p087'), (0.6067961165048543, 'p010'), (0.6048543689320388, 'p090'),
(0.6048543689320388, 'p056'), (0.6047156726768377, 'p097'), (0.596116504854369, 'p041'), (0.5825242718446602, 'p006'),
(0.577669902912621 3, 'p051'), (0.5755894590846047, 'p047'), (0.5714285714285714, 'p040'), (0.5711974110032362, 'p039'),
(0.5621821544151641, 'p037'), (0.5580212667591308, 'p045'), (0.5552473416551086, 'p021'), (0.5436893203883495, 'p071'),
(0.5427184466019417, 'p100'), (0.5395284327323162, 'p018'), (0.5388349514563107, 'p028'), (0.5385113268608415, 'p098'),
(0.5378640776699029, 'p062'), (0.5360610263522885, 'p085'), (0.5323624595469255, 'p029'), (0.5298196948682385, 'p052'),
(0.5242718446601942, 'p058'), (0.5242718446601942, 'p019'), (0.5194174757281553, 'p078'), (0.5148867313915857, 'p093'),
(0.5145631067961165, 'p096'), (0.5032362459546925, 'p032'), (0.5027739251040222, 'p075'), (0.5027739251040222, 'p033'),
(0.4907073509015257, 'p082'), (0.486084142394822, 'p072'), (0.48576051779935275, 'p036'), (0.4854368932038835, 'p034'),
(0.484604715672676 8, 'p091'), (0.4840499306518724, 'p073'), (0.47586685159500697, 'p076'), (0.47572815533980584, 'p031'),
(0.4728155339805825, 'p025'), (0.45987055016181233, 'p042'), (0.4592233009708738, 'p070'), (0.4429033749422099, 'p043'),
(0.4268608414239482, 'p002'), (0.4245954692556634, 'p069'), (0.41484049930651873, 'p065'), (0.4077669902912621, 'p048'),
(0.40614886731391586, 'p083'), (0.4058252427184466, 'p050'), (0.4058252427184466, 'p009'), (0.3943596856218215 5, 'p088'),
(0.39389736477115117, 'p080'), (0.39361997226074896, 'p023'), (0.38904299583911234, 'p049'), (0.38294036061026354, 'p092'),
(0.37309292649098474, 'p026'), (0.36893203883495146, 'p015'), (0.361997226074896, 'p001'), (0.35575589459084606, 'p030'),
(0.34951456310679613, 'p022'), (0.3424410540915395, 'p024'), (0.3300970873786408, 'm2'), (0.32940360610263525, 'p020'),
(0.31983356449375866, 'p089'), (0.31830790568654643, 'p084'), (0.30258899676375406, 'p086'), (0.3019417475728155, 'p011'),
(0.3016181229773463, 'p038'), (0.30097087378640774, 'p054'), (0.2912621359223301, 'p044'), (0.2815533980582524, 'p055'),
(0.2669902912621359, 'p095'), (0.25977808599167823, 'p064'), (0.22838650023116044, 'p027'), (0.22330097087378642, 'p079'),
(0.22330097087378642, 'p063'), (0.20970873786407768, 'p035'), (0.19648636153490523, 'p046'), (0.19514563106796118, 'p060'),
(0.1941747572815534, 'p004'), (0.17337031900138697, 'p008'), (0.16666666666666666, 'p003'), (0.1536754507628294, 'p059'),
(0.14563106796116504, 'p057'), (0.13592233009708737, 'p007'), (0.009708737864077669, 'm1')]

Thus our shortlist of 12 elements ['p013', 'p016', 'p017', 'p081', 'p053', 'p005', 'p067', 'p094', 'p066', 'p068', 'p014', 'p074']

7 Based on the previous best policies shortlist (see Question 6), what are your best-choice recommendations from the different objectives point of views and from a global multi-objectives compromise point of view ?

To solve this question, at first, I created a new partial performance tableau for each of the possible cases based on different points of views. Then, in the next step, I used the constructor *BipolarOutrankingDigraphs*() to create a new Bipolar Digraphs from that performance tableaux. Rubis solver helped me to get a potential Best Choice Recommendation also by calculating their credibility. RBCR is used to extract the best choices from subsets. Lets Show Rubis Best Choice Recommendation from the previously shortlisted set of historically motivated choices.

```
Objective under study performance
**********************
Rubis best choice recommendation(s) (BCR)
 (in decreasing order of determinateness)
Credibility domain: [-1.00,1.00]
 === >> potential best choice(s)
* choice            : ['p005', 'p013', 'p016', 'p017', 'p081']
  independence       : 0.00
  dominance          : 0.01
  absorbency         : -0.36
  covering (%)       : 42.86
  determinateness (%) : 52.35
  - most credible action(s) = { 'p081': 0.10, 'p005': 0.03, }

 === >> potential worst choice(s)
* choice            : ['p005', 'p053', 'p066', 'p067', 'p068', 'p074', 'p094']
  independence       : 0.00
  dominance          : -0.40
  absorbency         : 0.06
  covered (%)        : 42.86
  determinateness (%) : 53.74
  - most credible action(s) = { 'p005': 0.16, 'p074': 0.12, 'p094': 0.12, 'p053': 0.04, 'p067': 0.03, }
Execution time: 0.020 seconds
****************************
Objective under study environment
**********************
```

Rubis best choice recommendation(s) (BCR)
(in decreasing order of determinateness)
Credibility domain: [-1.00,1.00]
=== >> potential best choice(s)
* choice : ['p014', 'p068', 'p074']
independence : 0.00
dominance : 0.14
absorbency : -0.71
covering (%) : 70.37
determinateness (%) : 66.07
- most credible action(s) = { 'p068': 0.43, }

=== >> potential worst choice(s)
* choice : ['p013', 'p067']
independence : 0.00
dominance : -1.00
absorbency : 0.29
covered (%) : 90.00
determinateness (%) : 56.55
- most credible action(s) = { 'p013': 0.14, }

Execution time: 0.016 seconds
****
Objective under study economy
****
Rubis best choice recommendation(s) (BCR)
(in decreasing order of determinateness)
Credibility domain: [-1.00,1.00]
=== >> potential best choice(s)
* choice : ['p013', 'p053']
independence : 0.00
dominance : 0.33
absorbency : -1.00
covering (%) : 70.00
determinateness (%) : 73.61
- most credible action(s) = { 'p013': 0.67, }

=== >> potential worst choice(s)

* choice : ['p014', 'p017', 'p068', 'p094']

independence : 0.00

dominance : -1.00

absorbency : 0.33

covered (%) : 46.88

determinateness (%) : 55.56

- most credible action(s) = { 'p068': 0.33, }

Execution time: 0.014 seconds

****

Objective under study society

****

Rubis best choice recommendation(s) (BCR)

(in decreasing order of determinateness)

Credibility domain: [-1.00,1.00]

=== >> potential best choice(s)

* choice : ['p013', 'p017', 'p081']

independence : 0.20

dominance : 0.40

absorbency : -0.80

covering (%) : 88.89

determinateness (%) : 60.00

- most credible action(s) = { 'p081': 0.20, 'p017': 0.20, 'p013': 0.20, }

=== >> potential worst choice(s)

* choice : ['p068', 'p074']

independence : 0.00

dominance : -0.80

absorbency : 0.20

covered (%) : 95.00

determinateness (%) : 50.00

- most credible action(s) = { }

Execution time: 0.014 seconds

8 Explain with methodological arguments the differences, the case given, between your best choice recommenda- tions (see question 7) and your 5-best-ranked results obtained in the beginning (see question 3).

Answer :

The difference between results because of use different methods. Rubis best choice recommendation is based on the idea of outranking. Since Copeland and Rubis test both suspect p013 as a winner it's possible that it can be true economically and socially but we have to say that there is no better method neither better choice because for two alternatives *x* and *y*, *x* outranks *y* (*x*S*y*) if there is a significant majority of criteria which support an at least as good statement and there is no criterion which raises a veto against it.

So, some alternatives from our best ranked top 5 was simply outranked and, consequently, we have a bit different results. We can also notice the presence of p013 in Kohler and NetFlows results with a strong presence in economy.

It is important to always keep in mind that, based on pairwise outranking situations, there does not exist any unique optimal ranking; especially when we face such big data problems. Changing the number of quantiles, the component ranking rule, the optimized quantile ordering strategy, all this will indeed produce, sometimes even substantially, different global ranking results

Appendices :

1. Listing the content of Digraph3 using python.

```
import glob

path = 'home\Moad\Digraph3'


files = [f for f in glob.glob(path + "**/*.py", recursive=True)]

for f in files:
    print(f)
```

```
>>> for f in files:
...     print(f)
...
/home/Moad/Digraph3/arithmetics.py
/home/Moad/Digraph3/digraphs.py
/home/Moad/Digraph3/digraphsTools.py
/home/Moad/Digraph3/graphs.py
/home/Moad/Digraph3/htmlmodel.py
/home/Moad/Digraph3/linearOrders.py
/home/Moad/Digraph3/outrankingDigraphs.py
/home/Moad/Digraph3/perfTabs.py
/home/Moad/Digraph3/performanceQuantiles.py
/home/Moad/Digraph3/randomDigraphs.py
/home/Moad/Digraph3/randomNumbers.py
/home/Moad/Digraph3/randomPerfTabs.py
/home/Moad/Digraph3/setup.py
/home/Moad/Digraph3/sortingDigraphs.py
/home/Moad/Digraph3/sparseOutrankingDigraphs.py
/home/Moad/Digraph3/testLin.py
/home/Moad/Digraph3/transitiveDigraphs.py
/home/Moad/Digraph3/votingProfiles.py
/home/Moad/Digraph3/xmcda.py
/home/Moad/Digraph3/historicalData_23.py
/home/Moad/Digraph3/perTab_23.py
```

```
[Moad@ADT-23 Digraph3]$ ls
agrum                  docSphinx             korobovProjection12.png  outrankingDigraphs.py    randomDigraphs.py              test
arithmetics.py         examples              linearOrders.py          performanceQuantiles.py  randomNumbers.py               testLin.py
build                  gpl-3.0.txt           literature               perfTabs.py              randomPerfTabs.py              transitiveDigraphs.py
calmat                 gpl.txt               makefile                 perrinMIS                README                         uniSorting.xml
cython                 graphs.py             MANIFEST                 perTab_23.py             requirements.txt               votingProfiles.py
digraph3_copyright.html handbook             OldCython                __pycache__              setup.py                       xmcda
digraphs.py            historicalData_23.py  orientedGraph.dot        pyDoc                    sortingDigraphs.py             xmcda.py
digraphsTools.py       htmlmodel.py          orientedGraph.png        R                        sparseOutrankingDigraphs.py
[Moad@ADT-23 Digraph3]$
```

2. Create a VM on Linux. (Azure Cloud)

Accueil > Centre de démarrage rapide > Déployer une machine virtuelle >

Créer une machine virtuelle

Validation réussie

De base

Abonnement	Free Trial
Groupe de ressources	(nouveau) ADT-Project-23
Nom de la machine virtuelle	ADT-v7
Région	USA Centre Sud
Options de disponibilité	Aucune redondance d'infrastructure requise
Image	CentOS-based 8.1
Taille	Standard D2s v3 (2 processeurs virtuels, 8 Gio de mémoire)
Type d'authentification	Mot de passe
Nom d'utilisateur	Moadh
Ports d'entrée publics	SSH, HTTP, HTTPS
Spot Azure	Non

Créer   < Précédent   Suivant >   Télécharger un modèle pour automation

Votre déploiement a été effectué

Nom du déploiement : CreateVm-OpenLogic.CentOS-8_1-2020062...   Heure de début : 29/06/2020 à 19:44:37
Abonnement : 無料試用版   ID de corrélation : f708cd76-5ff7-4608-9a4c-cd4b6aecda13
Groupe de ressources : Moad-ADT

Détails du déploiement (Télécharger)

Ressource	Type	Statut	Détails de l'opération
ADT-23	Microsoft.Compute/virtual...	OK	Détails de l'opération
adt-23455	Microsoft.Network/network...	Created	Détails de l'opération
ADT-23-nsg	Microsoft.Network/network...	OK	Détails de l'opération
ADT-23-ip	Microsoft.Network/publicIp...	OK	Détails de l'opération

3. Showing the Instalation of some required packages (nose, r-tools and Azure- CLI)

```
[Moad@ADT-23 Digraph3]$ pip install nose
Defaulting to user installation because normal site-packages is not writeable
Collecting nose
  Downloading nose-1.3.7-py3-none-any.whl (154 kB)
     |████████████████████████████████| 154 kB 1.3 MB/s
Installing collected packages: nose
Successfully installed nose-1.3.7
[Moad@ADT-23 Digraph3]$
```

```
[Moad@ADT-23 Digraph3]$ R -version
WARNING: unknown option '-version'


R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

```
[Moad@ADT-23 ~]$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin
 and enter the code FB7TQCYZU to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "445a9c95-0f9d-4953-9db1-bc4a45dd1220",
    "id": "7d727df2-750b-4b81-b413-8db6c1ca6681",
    "isDefault": true,
    "managedByTenants": [],
    "name": "無料試用版",
    "state": "Enabled",
    "tenantId": "445a9c95-0f9d-4953-9db1-bc4a45dd1220",
    "user": {
      "name": "0190054306@uni.lu",
      "type": "user"
    }
  }
]
[Moad@ADT-23 ~]$
```

```
[Moad@ADT-23 ~]$ az interactive
This command is in preview. It may be changed/removed in a future release.
Installing the Interactive extension...
 - Installing ..
```

4. Sftp - Migrating my project from local machine to the VM.

Command Prompt - sftp Moad@52.247.232.125

```
C:\Users\chouaib>sftp Moad@52.247.232.125
The authenticity of host '52.247.232.125 (52.247.232.125)' can't be established.
ECDSA key fingerprint is SHA256:uOlvmlpr9u4ZXKqLThGkJOHrmn2meecbF/t75jYm1SA.
Are you sure you want to continue connecting (yes/no)?
Please type 'yes' or 'no':
Warning: Permanently added '52.247.232.125' (ECDSA) to the list of known hosts.
Moad@52.247.232.125's password:
Connected to Moad@52.247.232.125.
sftp> dir
Digraph3          Python-3.8.3      Python-3.8.3.tgz
sftp> pwd
Remote working directory: /home/Moad
sftp> lpwd
Local working directory: c:\users\chouaib
sftp> lcd c:\users\chouaib\Project_23
sftp> mput all
stat all: No such file or directory
sftp> mput a
stat a: No such file or directory
sftp> lcd c:\users\chouaib
sftp> mput Project_23
Uploading Project_23/ to /home/Moad/
Project_23/ is not a regular file
sftp> lcd c:\users\chouaib\Project_23
sftp> mput historicalData_23.py perTab_23.py README_23
Uploading historicalData_23.py to /home/Moad/perTab_23.py
historicalData_23.py                                         100% 2256KB  43.8KB/s   00:51
sftp> mput perTab_23.py README_23
stat perTab_23.py: No such file or directory
sftp> mput perfTab_23.py README_23
Uploading perfTab_23.py to /home/Moad/README_23
perfTab_23.py                                                 100%  118KB  38.1KB/s   00:03
sftp> mput README_23
Uploading README_23 to /home/Moad/README_23
README_23                                                     100% 2751    13.2KB/s   00:00
```

Note 1 : I use a nickname ‘chouaib’ as user for my laptop instead of my real name.

Note 2 : ‘mput’ help to move multiple files from a destination to another.

Note 3 : Sftp is just a choice, one among many others… FTPS. FTPS, known as FTP over SSL/TLS, is another option for businesses to employ for internal and external file transfers.

5. The Python Script afferent to Project 23.

```
from outrankingDigraphs import *

from sortingDigraphs import *

from linearOrders import *

# Extract the data

t = PerformanceTableau('perTab_23')


tableau = {'performance't,'economy':PartialPerformanceTableau(t, objectivesSubset=['Eco']),
'environment':PartialPerformanceTableau(t,objectivesSubset=['Env']),'society':PartialPerfor
manceTableau(t,objectivesSubset=['Soc']),'data': PerformanceTableau('historicalData_23')}


# Q2.

chordless =
BipolarOutrankingDigraph(tableau['performance']);chordless.computeChordlessCircuits();ch
ordless.showChordlessCircuits()


# Q3. Ranking RANKING_ALGORITHMS = [CopelandOrder, NetFlowsOrder, KohlerOrder]


  """ Strong and weak condorcet """

  print(digraph.condorcetWinners())

  print(digraph.weakCondorcetWinners())

  """ Apply ranking function on a digraph. """

  ranking = function(digraph)

  ranking.showRanking()

  correlation = digraph.computeOrdinalCorrelation(ranking)

  digraph.showCorrelation(correlation)

  """ Show the results from the ranking methods from the perspective

  of every objective. """

  for obj in objectives:

    print("Studying the perspective: {}.\n".format(obj))

    dg = BipolarOutrankingDigraph(tableau[obj])

    for f in ranking_algorithms:

      show_ranking(dg, f)

    condorcets(dg)

show_all_objectives(tab_data, RANKING_ALGORITHMS)
```

```
# Q 4 - Deciles Public Policy

quantile_sort = QuantilesSortingDigraph(tableau['performance'], 10)
quantile_sort.showQuantileOrdering(strategy='average')

#Q5 - Deciles Historical Record

# Absolute quantiles
pq = PerformanceQuantiles(tableau['data'], numberOfBins='deciles')
nqr = NormedQuantilesRatingDigraph(pq, tableau['performance'], rankingRule='best')

nqr.showHTMLRatingHeatmap()
nqr.showQuantilesRating() # The quantiles

#Q6.
pq = PerformanceQuantiles(tableau['data'], numberOfBins=3)
nqr = NormedQuantilesRatingDigraph(pq, tableau['performance'], rankingRule='best')
nqr.showQuantileSort()

#Q7.

shortlisted_tableau = {
    'performance': PartialPerformanceTableau(tableau['performance'],
actionsSubset=shortlist),
    'environment': PartialPerformanceTableau(tableau['environment'],
actionsSubset=shortlist),
    'economy'    : PartialPerformanceTableau(tableau['economy'], actionsSubset=shortlist),
    'society'   : PartialPerformanceTableau(tableau['society'], actionsSubset=shortlist),
}

for obj in shortlisted_tableau:
    print("Objective under study " + obj)
    dg = BipolarOutrankingDigraph(shortlisted_tableau[obj])
    dg.showRubisBestChoiceRecommendation()
```