Computing linear rankings from trillions of pairwise outranking situations

Raymond Bisdorff[1]

Abstract. We present in this paper a sparse HPC implementation for outranking digraphs of huge orders, up to several millions of decision alternatives. The proposed outranking digraph model is based on a quantiles equivalence class decomposition of the underlying multicriteria performance tableau. When locally ranking each of these ordered components, we may readily obtain an overall linear ranking of big sets of decision alternatives. For the local rankings, both, Copeland's as well as the Net-Flows ranking rules, appear to give the best compromise between, on the one side, the fitness of the overall ranking with respect to the given global outranking relation and, on the other side, computational tractability for very big outranking digraphs modelling up to several trillions of pairwise outranking situations.

1 Pre-ranked sparse outranking digraphs

When the preference learning community seams now well armed for tackling big preferential data [1], the multiple criteria decision aid community, and especially the one based on the outranking approach, still essentially tackles decision aid problems concerning only tiny or small sets of decision alternatives [2].

Challenged by the computational performances of the preference learning algorithms, we present in this paper a *sparse approximate model* of a given outranking digraph, which allows us, via HPC facilities, to efficiently rank very big sets of decision alternatives. Our starting point is the methodological consideration that an outranking digraph –the association of a set of decision alternatives and an outranking relation (see [3])– is, following the methodological requirements of the outranking based decision aid approach (see [4], [5], necessarily associated with a corresponding *multiple criteria performance tableau*. Such a tableau shows a given set of potential decision alternatives that are evaluated on a given *coherent family* (see Roy [3]) of weighted performance criteria.

In this paper we are going to use the preferential information delivered by such a given performance tableau[2] for linearly decomposing big sets of decision alternatives into ordered quantiles equivalence classes. This decomposition will lead us to a *pre-ranked, sparse approximate model* of an outranking digraph.

[1] University of Luxembourg, Faculty of Science, Technology and Communication, Computer Science and Communications Research Unit/ILIAS, url: http://leopold-loewenheim/bisdorff/ .

[2] Due to space limits we are not going to discuss here the actual elaboration of a performance tableau. We refer instead the reader to our recent book [2].

1.1 Showing the performance tableau

Strong motivation for trying to linearly rank a set of decision alternatives stems from the desire to show the corresponding performance tableau from a decision aiding perspective. Consider, for instance, the performance tableau showing the service quality of 12 commercial cloud providers graded by an external auditor on 14 ordinal, incommensurable performance criteria (see Wagle et al. [6]).

Example 1 (Service quality of commercial cloud providers)**.**

criterion	upT	dwT	ouT	LB	MTBF	Rcv	Lat	RspT	Thrpt	stoC	snpC	auT	enC	auD
Amz	2	2	2	4	3	3	NA	3	NA	4	NA	4	4	4
Cen	4	4	0	4	4	4	NA	2	NA	3	NA	4	4	4
Cit	2	4	2	4	3	4	NA	2	NA	3	4	4	4	4
Dig	2	1	4	4	3	3	NA	2	NA	3	NA	4	4	4
Ela	4	4	0	4	4	4	NA	4	NA	3	4	4	4	4
GMO	1	3	4	4	3	2	NA	4	NA	3	NA	4	4	4
Ggl	4	2	1	4	2	3	NA	2	NA	4	4	4	4	4
HP	3	3	2	4	4	3	NA	4	NA	3	4	4	4	4
Lux	2	2	2	4	3	3	NA	2	NA	2	NA	4	4	4
MS	4	4	0	4	4	4	NA	4	NA	4	NA	4	4	4
Rsp	NA	NA	NA	4	NA	3	NA	NA	NA	3	4	4	4	4
Sig	4	4	0	4	4	4	NA	3	NA	3	4	4	4	4

Legend: *0 = 'very weak', 1 = 'weak', 2 = 'fair', 3 = 'good', 4 = 'very good','NA' = missing data.*

Each row shows the ordinal grades that the auditor has provided for the respective cloud provider. Notice by the way the constant grades on some criteria and the many missing data. It is not evident to discover in this list who might be the potentially best performing cloud provider. The same performance tableau may better be linearly ranked from the best to the worst performing providers; ties, the case given, being resolved lexicographically.

Ranking of cloud providers by service quality

criteria	dwT	Rcv	MTBF	upT	RspT	stoC	auD	enC	auT	snpC	Thrpt	Lat	LB	ouT
weights	2.00	2.00	2.00	2.00	2.00	3.00	1.00	1.00	1.00	3.00	2.00	2.00	2.00	2.00
tau(*)	0.56	0.44	0.44	0.41	0.33	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.45
MS	4	4	4	4	4	4	4	4	4	NA	NA	NA	4	0
Ela	4	4	4	4	4	3	4	4	4	4	NA	NA	4	0
Sig	4	4	4	4	3	3	4	4	4	4	NA	NA	4	0
Cen	4	4	4	4	2	3	4	4	4	NA	NA	NA	4	0
HP	3	3	4	3	4	3	4	4	4	4	NA	NA	4	2
Cit	4	4	3	2	2	3	4	4	4	4	NA	NA	4	2
GMO	3	2	3	1	4	3	4	4	4	NA	NA	NA	4	4
Ggl	2	3	2	4	2	4	4	4	4	4	NA	NA	4	1
Rsp	NA	3	NA	NA	NA	3	4	4	4	4	NA	NA	4	NA
Amz	2	3	3	2	3	4	4	4	4	NA	NA	NA	4	2
Dig	1	3	3	2	2	3	4	4	4	NA	NA	NA	4	4
Lux	2	3	3	2	2	2	4	4	4	NA	NA	NA	4	2

Color legend:

quantile	20.00%	40.00%	60.00%	80.00%	100.00%

(*) tau: Ordinal (Kendall) correlation between marginal criterion and global ranking relation.

The grades observed on each criterion appear optimistically gathered in 5-tile equivalence classes. With 10 grades in the best quintiles class ($80\% - 100\%$), provider 'MS' appears here to be best performing, followed by provider 'Ela'. The criteria usually do not have the same weight in the decision problem. They appear ranked here in decreasing order of the ordinal correlation index[3] observed between the presentation ranking and the marginal criterion one.

This ranked presentation of a performance tableau is, without doubt, very useful for a decision aiding purpose, even more if the set of decision alternatives becomes bigger. But, how to rank now a big performance tableau gathering the evaluations of thousands or even millions of decision alternatives? The Copeland ranking rule, used for ranking the cloud providers above, is based on net crisp outranking flows which requires to compute the in- and out-degree of each node in the corresponding outranking digraph. When the order n of the outranking digraph now becomes big (several thousand or millions of decision alternatives), this ranking rule will require the handling of a huge set of n^2 pairwise outranking situations.

Yet, it is evident that, when facing such a big set of decision alternatives, the 20% best performing alternatives will for sure outrank the 20% worst performing ones. In a big data case, it may hence appear unnecessary to compute the complete set of the pairwise outranking situations among all decision alternatives. We shall therefore present hereafter a *pre-ranked, sparse approximate model* of the outranking digraph, where we only keep a linearly ranked list of quantiles performance classes with local outranking content.

1.2 Quantiles sorting of a performance tableau

To do so, we propose to decompose the given performance tableau into k ordered *quantiles equivalence classes*. Let X be the set of n potential decision alternatives evaluated on a single real performance criterion. We denote $x, y, ...$ the performances observed of the potential decision alternatives in X. We call *quantile* $q(p)$ the performance such that $p\%$ of the observed n performances in X are less or equal to $q(p)$. The quantile $q(p)$ is determined by the sorted distribution of the observed performances in X.

We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q+1$ equally spaced quantiles limits like *quartiles* limits: $0, .25, .5, .75, 1$, *quintiles limits*: $0, .2, .4, .6, .8, 1$, or *deciles* limits: $0, .1, .2, ..., .9, 1$. The *upper-closed* q^k *quantiles class* corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $p_q = \max_X x$ and the first class $q(p_1) =]-\infty; q(p_1)]$ gathers all data below $q(p_1)$. We call *q-tiles* a complete series of $k = 1, ..., q$ upper-closed q^k quantiles classes. Similarly, the *lower-closed* q_k quantiles class corresponds to the the interval $[q(p_{k-1}); q(p_k)[$, for $k = 1, ..., q-1$, where $p_1 = \min_X x$ and the last class $q(p_q) = [q(p_{k-1}); +\infty[$ gathers all data above $q(p_{k-1})$. We will in the sequel consider by default upper-closed quantiles.

If x is a measured performance on a single criterion, we may hence distinguish three sorting situations: $x \leqslant q(p_{k-1}) \equiv$ 'the performance x is lower than the q^k class'; $x > q(p_{k-1})$ and $x \leqslant q(p_k) \equiv$ 'the performance x belongs to the q^k class'; $x > q(p_k) \equiv$ 'the performance x is higher than the p^k class'. The relation $<$ being the *dual* of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

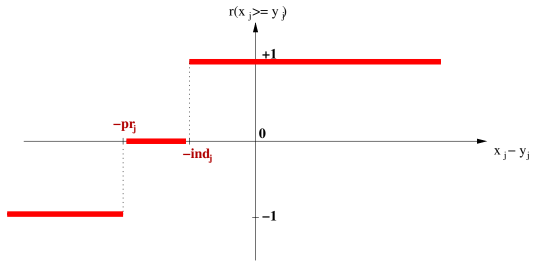
The multiple criteria extension of this single criterion q-sorting works as follows. Let $F = \{1, ..., m\}$ be a finite and coherent family of m performance criteria [4] and let x and y be two evaluation vectors on F. For each criterion j in F, we suppose the performances to be measured on a real scale $[0; M_j]$, supporting an indifference discrimination threshold ind_j, a preference discrimination threshold pr_j and a veto discrimination threshold v_j such that $0 \leqslant ind_j < pr_j < v_j \leqslant M_j$. The marginal evaluation of x on criterion j is denoted x_j. Each criterion j in F carries furthermore a *rational significance* weight w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

In the bipolar outranking approach (see Bisdorff 2013 [9]), every criterion $j \in F$ characterizes on X a marginal double threshold order $\geqslant_j$ with the following semantics (see Fig. 1):

$$r(x \geqslant_j y) \overset{\text{def}}{=} \begin{cases} +1 & \text{if} \quad x_j - y_j \geqslant -ind_j \\ -1 & \text{if} \quad x_j - y_j \leqslant -pr_j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$r(x \geqslant_j y) = +1$ signifies that x is *performing at least as good as* y on criterion j, $r(x \geqslant_j y) = -1$ signifies that x is *not performing at least as good as* y on criterion j, and $r(x \geqslant_j y) = 0$ signifies that it is *unclear* whether, on criterion j, x is performing at least as good as y. Each criterion j contributes thus, in the following way, the significance w_j of his marginal "*at least as good as*" characterization $r(x \geqslant_j y)$ to the global characterization $r(x \geqslant y)$:

Figure 1. Characteristic function of marginal '*at least as good as*' relation



$$r(x \geqslant y) \overset{\text{def}}{=} \sum_{j \in F} \left[\, w_j \cdot r(x \geqslant_j y) \, \right], \tag{2}$$

where: $r(x \geqslant y) > 0$ signifies that x is *globally performing at least as good as* y; $r(x \geqslant y) < 0$ signifies that x is *not globally performing at least as good as* y; and, $r(x \geqslant y) = 0$ signifies that it is *unclear* whether x is globally performing at least as good as y.

From an epistemic point of view, we say that evaluation x *outranks* evaluation y, denoted $(x \succsim y)$, if a *significant majority of criteria validates* a global outranking situation between x and y, i.e. $(x \geqslant y)$ and *no veto* (denoted $(x \not\lll y)$) is observed on a discordant criterion. Similarly, evaluation x *does not outrank* evaluation y, denoted $(x \not\succsim y)$, if *a significant majority of criteria invalidates* a global outranking situation between x and y, i.e. $(x \not\geqslant y)$ and *no counter-veto* (denoted $(x \not\ggg_j y)$) is observed on a concordant criterion j [9]. Veto characteristics on a criterion j (denoted $(x \lll_j y)$) are defined as follows:

$$r(x \lll_j y) \overset{\text{def}}{=} \begin{cases} +1 & \text{if} \quad x_j - y_j \leqslant -v_j \\ -1 & \text{if} \quad x_j - y_j \geqslant v_j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

If we furthermore set $r(x \ggg_j y) \overset{\text{def}}{=} r(x \not\lll_j y)$, veto and counter-veto situations will be *codual* one to another.

Now, a global bipolar outranking characteristic $r(x \succsim y)$ is defined as follows:

$$r(x \succsim y) \overset{\text{def}}{=} r(x \geqslant y) \oslash \left(\oslash_{j \in F} \left[r(x \not\lll_j y) \right] \right).^{5} \tag{4}$$

[3] Extended Kendall τ index, see Bisdorff 2012 [7].

[4] Coherent means here: *universal*, *minimal* and *preferentially consistent wrt marginal preferences* (see Roy 1991, 1993 [3, 8] and Bisdorff 2002 [5]).

In particular: $r(x \succsim y) = r(x \geqslant y)$ if no considerable large positive or negative performance differences are observed; $r(x \succsim y) = 1$ if $r(x \geqslant y) \geqslant 0$ and $\circledvee_{j \in F}[r(x \ggg_j y)] = 1$; and, $r(x \succsim y) = -1$ if $r(x \geqslant y) \leqslant 0$ and $\circledvee_{j \in F}[r(x \lll y)] = 1$.

For $k = 1, ..., q$, let $\mathbf{q^k}$ denote a multiple criteria quantiles class, and let $\mathbf{q}(p_{k-1}) = \big(q_1(p_{k-1}), q_2(p_{k-1}), ..., q_m(p_{k-1})\big)$ denote its *lower limits*, and $\mathbf{q}(p_k) = \big(q_1(p_k), q_2(p_k), ..., q_m(p_k)\big)$ its corresponding *upper limits*. Alternative x belongs to multiple criteria quantiles class $\mathbf{q^k}$ if the corresponding lower limits $\mathbf{q}(p_{k-1})$ *do not outrank*, whereas the upper limits $\mathbf{q}(p_k)$ *do outrank* the multiple criteria performances of x. The membership characteristic function evaluating if of alternative x belongs to quantiles class $\mathbf{q^k}$ may be readily assessed as follows:

$$r(x \in \mathbf{q^k}) \overset{\text{def}}{=} \min\big[\, -r\big(\mathbf{q}(p_{k-1}) \succsim x\big),\; r\big(\mathbf{q}(p_k) \succsim x\,\big)\,\big] \quad (5)$$

In our bipolar characteristic calculus, *logical conjunction* is indeed implemented via the min operator, whereas *logical negation* is implemented by changing the sign of the r values (see [4, 5, 9]).

Formula 5 gives us now the essential tool for implementing our q-tiles sorting algorithm.

The multicriteria (upper-closed) q-tiles sorting algorithm

```
Input: a set $X$ of $n$ decision alternatives with a performance
tableau on a family of $m$ criteria and a set $\mathcal{Q} = \{\mathbf{q^1}, ..., \mathbf{q^q}\}$
of $k = 1, .., q$ empty multiple criteria quantiles classes.
For each object $x \in X$ and each quantiles class $\mathbf{q^k} \in \mathcal{Q}$
  1. $r(x \in \mathbf{q^k}) \leftarrow \min\big(-r(\mathbf{q}(p_{k-1}) \succsim x), r(\mathbf{q}(p_k) \succsim x)\big)$
  2. if $r(x \in \mathbf{q^k}) \geqslant 0$ add $x$ to $q$-tiles class $\mathbf{q^k}$
Output: $\mathcal{Q}$
```

The complexity of the q-tiles sorting algorithm is in $\mathcal{O}(nmq)$, i.e. *linear* in the number of decision alternatives (n), criteria (m) and quantiles classes (q). As $\mathcal{Q}$ represents a partition of the criterion performance measurement scales, there is a potential for run time optimization.

We may compute a didactic example with the help of our DIGRAPH3 collection of Python modules. [6]

Example 2 (Python session with DIGRAPH3 resources)

```
1 >>> from randomPerfTabs import RandomPerformaceTableau
2 >>> t = RandomPerformanceTableau(numberOfActons=50,
  ...                                    seed=5)
3 >>> from sparseOutrankingDigraphs import\
  ...                  PreRankedOutrankingDigraph
4 >>> pr = PreRankedOutrankingDigraph(t,quantiles=5)
5 >>> pr.showSorting()
*--- Sorting results in descending order ---*
]0.8 - 1.0]:  [a16, a2, a24, a32]
]0.6 - 0.8]:  [a01, a02, a06, a09, a10, a13, a16, a18,
               a22, a25, a27, a28, a31, a32, a36, a37,
               a39, a40, a41, a43, a45, a48]
]0.4 - 0.6]:  [a01, a03, a04, a05, a07, a08, a09, a10,
               a11, a12, a13, a14, a15, a17, a18, a20,
               a26, a27, a29, a30, a33, a34, a35, a38,
               a42, a43, a44, a45, a46, a47, a49, a50]
]0.2 - 0.4]:  [a04, a11, a12, a17, a19, a21, a23,
               a29, a34, a42, a46, a47, a50]
] < - 0.20]:  []
```

In Lines 1 and 2 we import a random performance tableau generator class and construct a sample '*RandomPerformanceTableau*' instance called t. In Lines 3 we import the '*PreRankedOutrankingDigraph*' class and in Line 4 we construct a 5-tiles sorting digraph called pr. Line 5 shows the actual contents of the quintiles performance classes we obtain with this random performance tableau instance. Notice the seed=5 argument in Line 2 which makes the random experiment repeatable.

As made evident with this example of 5-tiling, useful formal properties of our q-tiles sorting algorithm are the following:

1. *Coherence*: Each decision alternative is always sorted into a non-empty subset of adjacent q-tiles classes. For instance, alternative 'a16' is sorted into the best ($]0.8 - 1.0]$) and second best quintiles class ($]0.6 - 0.8]$). In the limit, a not yet evaluated alternative would appear sorted in all five quintiles classes.
2. *Uniqueness*: If no indeterminate outranking situation is being observed ($r() \neq 0$), a decision alternative is sorted into exactly one single quantiles class. This is the case, for instance, with alternative 'a24' which is solely sorted into the best quintiles class ($]0.8 - 1.0]$).
3. *Separability*: The quantiles class limits $\mathbf{q}(p^k)$ being given, the sorting result for each alternative x may be computed independently of the other alternatives' sorting results. Similarly, the content of a quantiles class $\mathbf{q^k}$ may be computed independently of the other classes' contents.

The last property will give us later on access to efficient *parallel processing* of class membership characteristics $r(x \in \mathbf{q^k})$ for all $x \in X$ and $k = 1, ..., k$.

1.3 A pre-ranked sparse approximation of the global outranking relation

Following the coherence property above, we may compute for each alternative x in X a *lower* and an *upper* q-tiles sorting limit. The lower limit is determined by the one of its lowest q-tiles class whereas the upper limit is determined by the one of its highest q-tiles class.

Reconsidering the quintiles sorting result of Example 2, we may observe, for instance, a decomposition of X into seven quantiles performance classes:

```
>>> pr.showDecomposition()
*--- quantiles decomposition in decreasing order---*
    c1. ]0.8-1.0]: [a24]
    c2. ]0.6-1.0]: [a16,a22, a32]
    c3. ]0.6-0.8]: [a02, a06, a25, a28, a31, a36, a37,
                    a39,  a40, a41, a48]
    c4. ]0.4-0.8]: [a01, a09, a10, a13, a18,
                    a27, a43, a45]
    c5. ]0.4-0.6]: [a03, a05, a07, a08, a14, a15, a20,
                    a26, a30, a33, a35, a38, a44, a49']
    c6. ]0.2-0.6]: [a04,'a11, a12, a17, a29, a34, a42,
                    a46, a47, a50]
    c7. ]0.2-0.4]: [a19, a21, a23]
```

Alternative 'a24', for instance, is sorted into the qantiles class $]0.8 - 1.0]$, whereas, alternative 'a16' is sorted into the quantiles class $]0.6 - 1.0]$.

The q-tiles sorting result leaves us hence with a partition of the set of decision alternatives into more or less refined quantiles performance classes. These classes may furthermore be linearly ranked from best to worst by following three potential ranking strategies:

1. *Optimistic*: In decreasing lexicographic order of, first, the upper and, secondly, the lower quantile;
2. *Pessimistic*: In decreasing lexicographic order of, first, the lower and, secondly, the upper quantile;

[5] $\circledvee$ denotes the symmetric or epistemic disjunction operator. If ϕ and φ denote two propositions, $r(\phi \circledvee \varphi)$ will be $\max(r(\phi), r(\varphi))$ if both $r(\phi)$ and $r(\varphi)$ are positive; $\min(r(\phi), r(\varphi))$ if both $r(\phi)$ and $r(\varphi)$ are negative; and 0 if $r(\phi)$ and $r(\varphi)$ are of opposite signs.

[6] Tutorials for programming with the DIGRAPH3 Python3 resources (see [10]) may be found on this site: http://leopold-loewenheim.uni.lu/docDigraph3/ .

3. *Average*: In decreasing lexicographical order of, first, the average of the lower and upper quantile, and secondly, the upper quantile.

Practical experiments with ranking given performance tableaux, like the one in Example 1 above, suggest that the '*average*' ranking strategy gives the most convincing result when indeterminate outranking situations and/or missing data are observed.

In view of the partition $\mathcal{P}_q = \{c_1, ..., c_k\}$ of the set X of decision alternatives, ranked from the best to the worst, we may define as follows the characteristic function of what we will call a *pre-ranked sparse* or *q-tiled* outranking relation, denoted $\gtrapprox^q$:

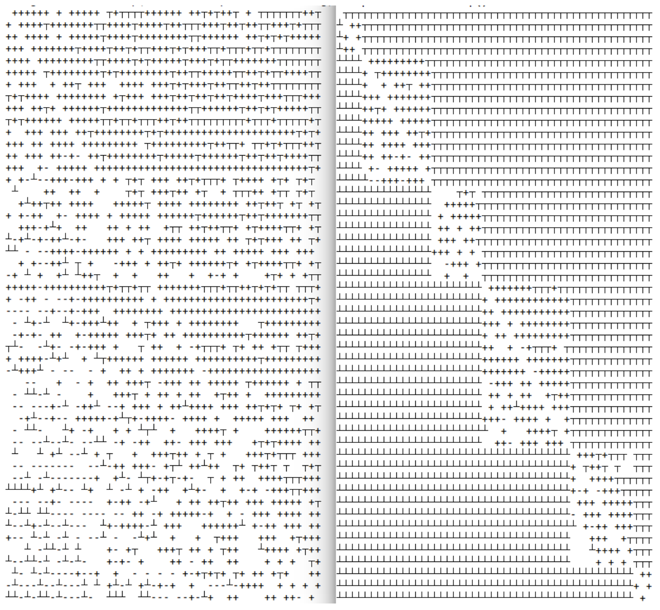
$$r(x \gtrapprox^q y) \overset{\text{def}}{=} \begin{cases} +1, & \text{if } \big[x \in c_i \wedge y \in c_j \wedge i < j\big] \\ -1, & \text{if } \big[x \in c_i \wedge y \in c_j \wedge i > j\big] \\ r(x \succsim_{|c_i} y) & \text{, otherwise.} \end{cases} \tag{6}$$

Relation $\succsim_{|c_i}$ denoting the restriction of the global outranking relation to the component c_i, $r(x \gtrapprox^q y) = r(x \succsim_{|c_i} y)$ only when x and y do appear in the same component c_i. The corresponding outranking digraph, denoted $G(X, \gtrapprox^q)$, is called a *pre-ranked* or *q-tiled* outranking digraph.

Depending on the number q of quantiles used in the q-tiles sorting algorithm, a more or less refined pre-ranking is obtained. In the limit, when, on the one hand, only one single component $c_1 = X$ is observed, we recover the standard outranking relation $\succsim$. On the other hand, when q is high and n singleton quantiles classes are obtained, we directly recover a linear ranking of the decision alternatives.

Reconsidering the performance tableau of Example 2, concerning a set of 50 decision alternatives, we show below a map of the standard outranking relation $\succsim$ (see Fig. 2. left side) versus a map of the corresponding pre-ranked outranking relation $\gtrapprox^5$ (see Fig. 2. right side).

Figure 2. Map of standard $\succsim$ outranking relation (left) versus corresponding pre-ranked, sparse approximate $\gtrapprox^5$ outranking relation (right) [8]



[8] Fig. 2. symbols legend: $\top \equiv$ outranking *for certain*; + $\equiv$ more or less *outranking*; ' ' $\equiv$ *indeterminate*; - $\equiv$ more or less *outranked*; $\perp \equiv$ outranked *for certain*.

The 5-tiled outranking relation of Example 2 shows indeed seven ordered components with a minimal cardinality of 1 (component c_1) and a maximal cardinality of 14 (component c_5). The outranking *fill rate* of the pre-ranked $\gtrapprox^5$ outranking relation, i.e. the actual remaining part in $\gtrapprox^5$ of the complete standard outranking relation $\succsim$ (see Fig. 2), amounts here to 18%. And the ordinal correlation index (see [7]) between the standard ($\succsim$) and the pre-ranked ($\gtrapprox^5$) outranking relation, denoted $\tau(\succsim, \gtrapprox^5)$ is, in this case, $+0.75$.

2 Computing linear rankings from a pre-ranked sparse outranking digraph

The previous quantiles sorting result represents in fact a first step in the construction of a global linear ranking of all given decision alternatives.

2.1 Heuristic linear closures of pairwise outranking situations

To linearly rank indeed the complete set X of decision alternatives, we still need to locally rank without ties the k components $c_i \subseteq X$ for $i = 1, ..., k$. To do so, we will rely on the component-wise restricted pairwise valued outranking situation characteristics, i.e. $r(x \succsim_{|c_i} y)$ for all x, y in c_i. We denote $\mathcal{G}_{|c_i}$ the restriction of the standard outranking digraph to the subset c_i of decision alternatives.

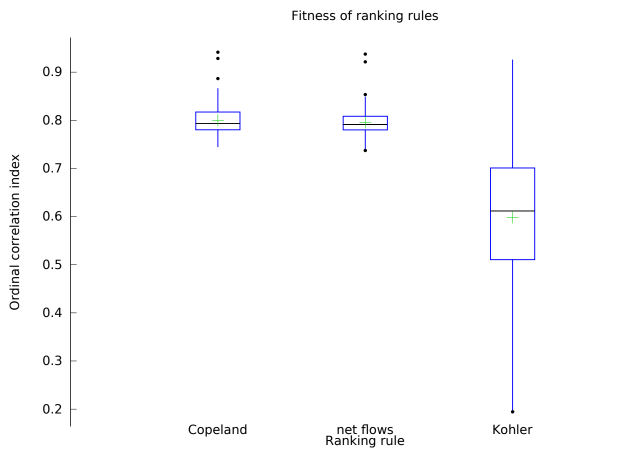
The component-wise ranking algorithm

```
1. Input: the outranking digraph $\mathcal{G}(X, \gtrapprox^q)$, a partition $P_q = \{c_1, c_2, ..., c_k\}$ of $k$ linearly ordered decreasing parts of $X$ obtained by the $q$-tiles sorting algorithm, and an empty list $\mathcal{R}$.
2. For each performance class $c_i \in P_q$:
      if $\#(c_i) > 1$:
      $R_k \leftarrow$ locally rank $c_i$ in $\mathcal{G}_{|c_i}$ with Copeland's rule
      (if ties, render the concerned alternatives in alphabetic order)
      else: $R_k \leftarrow c_i$
      append $R_k$ to $\mathcal{R}$
3. Output: $\mathcal{R}$
```

The complexity of the component-wise ranking algorithm is *linear* in the number k of components resulting from a q-tiles sorting. Concerning the compexity of the local ranking procedure, we know that outranking relations do only exceptionally show linear rankings. Usually, they do not even render complete or, at least, transitive partial relations (see Bouyssou and Pirlot 2005 [11]). Three heuristic ranking rules appear most suitable here for our purpose –*Copeland*'s, *Net-flows*' and *Kohler*'s rule– all three of complexity $\mathcal{O}(\#(c_i)^2)$ on each restricted outranking digraph $\mathcal{G}_{|c_i}$. In case of *ties* (very similar evaluations for instance), the local ranking procedure will render these alternatives in increasing *alphabetic ordering* of their identity keys.

However, the three considered ranking rules do not deliver rankings of a same quality as we show in Fig. 3 below. We may notice, indeed, that the quality of the linear rankings obtained with Copeland's or the Net-Flows ranking rule on a sample of 100 outranking digraphs of order 250 is much better (median correlation with $\succsim$ around $+0.8$ [7]) when compared with Kohler's rule (median correlation with $\succsim$ around $+0.6$ only). As Copeland's ranking rule is the simplest to implement, we will by default use this ranking rule in the sequel.

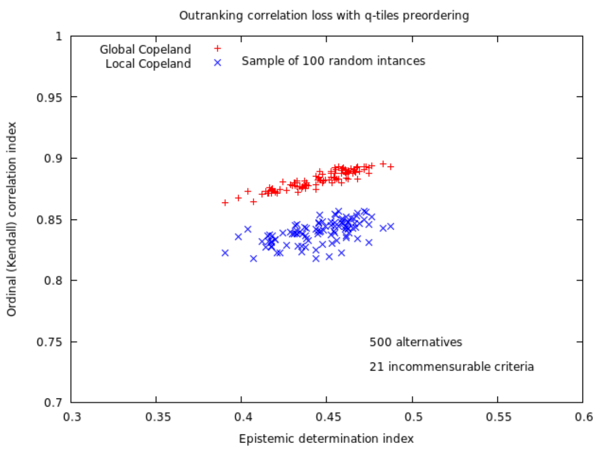
Figure 3. Sample of hundred 10-tiled outranking graphs of order 250



2.2 Fitness of the linear ranking result

The fitness of our pre-ranked sparse versus the standard outranking digraph model may be appreciated when comparing both the ordinal quality of the linear ranking result obtained from a standard versus the result obtained from a 50-tiled pre-ranked outranking relation. We consider a sample of 100 random performance tableaux gathering 500 decision alternatives evaluated on 21 incommensurable performance criteria. The 50-tiles pre-rankings give on average around 38 (sd. 7.6) performance classes, with a minimum of 32 and a maximum of 67 components, leading to an average diagonal outranking fill rate of around 3% (sd: 0.4%). Concerning the ordinal correla-

Figure 4. Standard versus 50-tiled pre-ranked sparse outranking model



tion (see [7]) between the standard outranking relation and our linear ranking result, we may notice in Fig. 4 that the correlation is around $+0.88$ (sd: 0.007) when the Copeland ranking is constructed from the global outranking relation, and around $+0.83$ (sd: 0.009), when it is constructed from the 50-tiled sparse outranking relation. The pre-ranking step, hence, does apparently not deteriorate significantly the quality of the ranking result.

Notice furthermore in Fig. 4 that the ordinal correlations are, both times, augmenting with the epistemic determination index of the standard outranking relation (see [7]). It is indeed evident that the quality of any linear ranking result essentially depends on the actual performance evaluations. When, on the one hand, they are very similar or there are many missing data and/or vetoes, the asymmetric part of the outranking relation will appear weakly determined. All potential ranking rules will deliver rankings that are more or less *weakly* correlated with the corresponding outranking relation. On the other hand, if the evaluations show a clearly determined alignment of the decision alternatives, the outranking relation will appear strongly determined, and all potential ranking rules will produce more or less *highly* correlated linear rankings.

2.3 Multithreading the q-tiles sorting & ranking procedure

When tackling big performance tableaux, evaluating thousands or even millions of decision alternatives, we absolutely need to speed up the computations with multiple parallel threading HPC implementations. This is readily made possible by the *separability* property of the pairwise outranking approach.

Relying indeed on this property when sorting each alternative into its respective quantiles performance classes, the q-tiles sorting algorithm may be *safely split* into as much parallel processing threads as there are *parallel processing units* available. Notice that a high number of parallel processing units, sharing a same CPU memory, will consequently need a very big memory. Furthermore, the component-wise ranking procedures, being all local to a diagonal component c_i, may as well be safely processed in *parallel threads* on each restricted outranking digraph $\mathcal{G}_{|c_i}$.

Along these ideas, we have specially adapted our DIGRAPH3 computing resources[9] in order to run our q-tiles sorting & ranking algorithms on the HPC clusters of the University of Luxembourg (see [12]). The computations reported here were operated on the *gaia-80* node, a Delta D88x-M8-BI, 8 * Intel Xeon E7-8880 v2 @ 2.5 GHz machine equipped with 120 single threaded processing cores and a shared CPU memory of 3 TB.

The multithreaded versions of our algorithms are implemented with the help of the Python3.5 `multiprocessing` module[10] and our DIGRAPH3 collection of Python3 modules [10]. We use, for instance, the following generic algorithmic design for implementing parallel local ranking procedures.

Generic approach for parallel processing of the local rankings

```
from multiprocessing import Process, active_children
    class Thread(Process):
        def __init__(self, threadID, localComp)
                    Process.__init__(self)
                    self.threadID = threadID
                    self.localComponent
                    ...
        def run(self):
            ... Copeland ranking
            ... of self.localComponent
            ...

nbrOfJobs = number of // CPUs
for job in range(nbrOfJobs):
    ... pre-threading tasks per job
    print('iteration = ',job+1,end=" ")
    splitThread = Thread(job, localComponent, ...)
    splitThread.start()
while active_children() != []:
    pass
print('Exiting parallel threads')
for job in range(nbrOfJobs):
    ... post-threading tasks per job
```

In Table 1 we show run times of linear ranking constructions both, from a standard $\succsim$ outranking relation and, from a pre-ranked, sparse approximate $\gtrsim^q$ outranking relation.

[9] See the documentation of the DIGRAPH3 resources FSTC/ILIAS Decision Systems Group, University of Luxembourg. http://leopold-loewenheim.uni.lu/docDigraph3 .

[10] See https://docs.python.org/2/library/multiprocessing.html .

Table 1. Comparing the standard and pre-ranked sparse approach[11]

digraph	$\succsim$ relation		$\gtrapprox^q$ relation	
order	run time	$\tau(>, \succsim)$	run time	$\tau(>^q, \succsim)$
1 000	6"	$+0.88$	4"	$+0.83$
2 000	15"	$+0.88$	9"	$+0.83$
2 500	27"	$+0.88$	14"	$+0.83$

If the speed gain for order 1 000 is 33%, we already reach nearly 50% acceleration with order 2 500. Notice that the quality, in terms of the correlation index with the gobal outranking relation, of the linear ranking result appears to be independent of the actual order, namely $+0.88$ for the standard, resp $+0.83$ for the pre-ranked outranking relation.

These excellent run times encouraged us to furthermore develop specific cythonized[12] C versions of our DIGRAPH3 Python modules in order to tackle pre-ranked outranking digraphs with sizes up to five trillions (5×10^{12}) of pairwise outranking situations (see Table 2). For the biggest instances, we generate a random 3 decision objectives performance tableau with 13 incommensurable criteria and 5% missing data.[13] For 1 000 000 alternatives (input data size = 1.4 GB) we thus obtain (see Table 2), with a 1 100-tiles pre-ranking, 13 547 diagonal components with a minimal size of 10 and a maximal size of 1 150 alternatives, leading to a fill rate of 0.05%. To linearly rank this huge digraph of size 10^{12} we need, on the gaia-80 machine with 120 parallel processing cores, in total about 1 hour and 17 minutes.

Table 2. HPC performance measurements for big data

$\gtrapprox^q$ outranking relation		q	fill	run
order	size		rate	time
10 000	1×10^8	150	0.64%	13"
15 000	2.25×10^8	200	-	22"
25 000	6.25×10^8	300	-	39"
50 000	2.5×10^9	500	0.58%	2'
100 000	1×10^{10}	900	0.22%	5'
1 000 000	1×10^{12}	1100	0.05%	1h17'
1 732 051	3×10^{12}	1500	0.04%	3h09'
2 236 068	5×10^{12}	1600	0.03%	4h50'

Notice that the choice of the number q of quantiles is of crucial importance for our computations as it influences both the run times of the q-tiling as well as that of the local rankings. If q is relatively small, there will be less components, making on the one hand, the q-tiles sorting procedure quicker. Yet, some components might in consequence show much larger cardinalities, which make, on the other hand, these local ranking procedures quadratically slower. If q is chosen larger, q-tiling will get linearly slower. Yet, the local rankings might get much quicker in case there appear less components of larger cardinalities. Best overall run times are obtained in practice with a number q of quantiles that makes the q-tiling procedure take approximately the same run time as the local rankings.

Finally, we estimated the quality of such huge linear ranking, denoted $>_q$, by sampling the ranking quality on sub-digraphs of order 1 000 for instance. Here we recovered in fact, by the vertu of the LLN, an average sampled correlation result we have already noticed in Table 1 , namely $\overline{\tau}(\succsim, >_q) \leadsto +0.83$, with a standard deviation diminishing with the growth of the number of samples we take into account. This result confirms again that, given the same random performance tableau generator, the quality of our ranking does not depend on the actual order and size of the outranking digraph.

[11] Legend: $\tau(>, \succsim)$, resp. $\tau(>^q, \succsim)$ show the ordinal correlations between the corresponding linear ranking results, denoted $>$, resp. $<^q$, and the given standard outranking relation $\succsim$.

[12] Cython: C-extensions for Python: http://cython.org/ .

[13] On generating random performance tableaux. See Tutorials of the DIGRAPH3 ressources .

Conclusion

We present in this paper a sparse, approximate outranking digraph model coupled with a two steps ranking algorithm based on quantiles-sorting & local-ranking procedures. Ranking results obtained with this outranking digraph model fit well with those given by a standard outranking digraph. Furthermore, separable quantiles sorting and local ranking procedures offer effective multiprocessing capacities. Efficient *scalability* allows, hence, the linearly ranking of very big performance tableaux gathering up to millions of evaluations. Good perspectives for further optimization with cython C implementations and HPC ad hoc tuning are given. All Python and cython HPC modules are freely available for further developments on the git repository of the DIGRAPH3 resources: http://github.com/rbisdorff/Digraph3 .

Acknowledgments. The author would like to thank the UL-HPC administration team for their helpful and competent assistance in the fine tuning of our HPC work. The anonymous reviewers, with their pertinent comments and suggestions, also much helped enhancing content and readability of this paper.

REFERENCES

[1] N. Japkowicz and J. Stefanowski (2015). *Big Data Analysis: New Algorithms for a New Society*. Springer-Verlag Berlin Heidelberg, Studies in Big Data, 329 pages.

[2] R. Bisdorff, L.C. Dias, P. Meyer, V. Mousseau and M. Pirlot (Eds.) (2015). *Evaluation and decision models with multiple criteria: Case studies*. Springer-Verlag Berlin Heidelberg, International Handbooks on Information Systems, DOI 10.1007/978-3-662-46816-6-1, 643 pages.

[3] B. Roy (1991). The outranking approach and the foundations of the Electre methods. *Theory and Decision*, 31(1):49–73.

[4] R. Bisdorff (2000). Logical foundation of fuzzy preferential systems with application to the electre decision aid methods. *Computers and Operations Research* (Elsevier) 27:673–687.

[5] R. Bisdorff (2002). Logical Foundation of Multicriteria Preference Aggregation. In: Bouyssou D et al (eds) *Aiding Decisions with Multiple Criteria*. Kluwer Academic Publishers, pp 379–403.

[6] S. S. Wagle, M. Guzek, P. Bouvry and R. Bisdorff (2015). An Evaluation Model for Selecting Cloud Services from Commercially Available Cloud Providers. In Proceedings of the 7th IEEE *International Conference on Cloud Computing Technology and Science*, Vancouver, Canada, November 30 - December 2 2015, ISBN 978-1-4673-9560-1/15, pp 107–114.

[7] R. Bisdorff (2012). On measuring and testing the ordinal correlation between bipolar outranking relations. In Proceedings of DA2PL'2012 - *From Multiple Criteria Decision Aid to Preference Learning*. University of Mons, November 15-16, pp 91–100.

[8] B. Roy and D. Bouyssou (1993). *Aide Multicritère à la Décision : Méthodes et Cas*. Economica Paris, 695 pages.

[9] R. Bisdorff (2013) On polarizing outranking relations with large performance differences. *Journal of Multi-Criteria Decision Analysis* (Wiley) 20:3-12.

[10] R. Bisdorff (2016). Tutorials of the DIGRAPH3 resources. *Documentation of the* DIGRAPH*3 resources* FSTC/ILIAS Decision Systems Group, University of Luxembourg. http://leopold-loewenheim.uni.lu/docDigraph3 .

[11] D. Bouyssou and M. Pirlot (2005). A characterization of concordance relations. *European Journal of Operational Research* 167: 427–443.

[12] S. Varrette, P. Bouvry, H. Cartiaux and F. Georgatos (2014). Management of an Academic HPC Cluster: The UL Experience. In Proc. of the 2014 Intl. Conf. on *High Performance Computing & Simulation* (HPCS 2014), Bologna (Italy). IEEE, pp 959–967.