Computing linear rankings from trillions of pairwise outranking situations

Raymond Bisdorff[1]

Contents

Abstract. We present a sparse HPC implementation for big outranking digraphs of huge orders, up to several millions of decision alternatives. Our sparse 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 obtain an overall linear ranking of the complete given set of decision alternatives. Both, Copeland's as well as the Net-Flows ranking rules appear herefore 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

Our starting point is the consideration that an outranking digraph – the association of a set of decision alternatives and an outranking relation– is, following the methodological requirements of the outranking approach, necessarily associated with a corresponding performance tableau [1, 2, 5]. Here we are going to use this underlying performance tableau for linearly decomposing big sets of decision alternatives into ordered quantiles equivalence classes. This decomposition will lead to a pre-ranked sparse outranking digraph.

1.1 Showing the performance tableau

Consider, for instance, a performance tableau showing the service quality of 12 commercial cloud providers measured by an external auditor on 14 incommensurable performance criteria [3].[2]

Performance table auditor2_1

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; '*green*' and '*red*' mark the **best**, respectively the **worst**, performances on each criterion.

The same performance tableau may be linearly ordered from the best to the worst performing alternatives (ties are lexicographically resolved), following for instance the Copeland ranking rule applied to the corresponding outranking digraph and eventually presented like a heat map coloured with the highest *7-tiles* class of the marginal performances:

Heatmap of Performance Tableau 'auditor2_1'

criteria	stoC	snpC	upT	dwT	ouT	LB	MTBF	Rcv	Lat	RspT	Thrpt	auT	enC	auD
weights	3.00	3.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00	1.00
MS	4	NA	4	4	0	4	4	4	NA	4	NA	4	4	4
Ela	3	4	4	4	0	4	4	4	NA	4	NA	4	4	4
Sig	3	4	4	4	0	4	4	4	NA	3	NA	4	4	4
Cen	3	NA	4	4	0	4	4	4	NA	2	NA	4	4	4
HP	3	4	3	3	2	4	4	3	NA	4	NA	4	4	4
Cit	3	4	2	4	2	4	3	4	NA	2	NA	4	4	4
Ggl	4	4	4	2	1	4	2	3	NA	2	NA	4	4	4
GMO	3	NA	1	3	4	4	3	2	NA	4	NA	4	4	4
Rsp	3	4	NA	NA	NA	4	NA	3	NA	NA	NA	4	4	4
Amz	4	NA	2	2	2	4	3	3	NA	3	NA	4	4	4
Dig	3	NA	2	1	4	4	3	3	NA	2	NA	4	4	4
Lux	2	NA	2	2	2	4	3	3	NA	2	NA	4	4	4

Color legend:

quantile	0.14%	0.29%	0.43%	0.57%	0.71%	0.86%	1.00%

But, how to rank a big performance table gathering the evaluations of thousands or even millions of decision alternatives? The Copeland ranking rule mentioned before is based on crisp net flows requiring to compute the in- and out-degree of each node in the outranking digraph. When the order n of the outranking digraph becomes big (several thousand or millions of decision alternatives), this ranking

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

[2] 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.

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 10% best performing alternatives will for sure outrank the 10% worst performing ones. In the big data case, it appears hence unnecessary to compute the complete set of the pairwise outranking situations among these two subsets of decision alternatives, for instance. We shall therefore present hereafter a *pre-ranked sparse model* of the outranking digraph, where we only keep a linearly ranked list of equivalent performance classes with local outranking content.

1.2 Quantiles sorting of a performance table

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 estimated by *linear interpolation* from the cumulative distribution of the performances in X.

We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q+1$ equally spaced quantiles limits like *quartile* limits: $0, .25, .5, .75, 1$, *quintile limits*: $0, .2, .4, .6, .8, 1$, or *decile* limits: $0, .1, .2, ..., .9, 1$ limits. The *upper-closed* q^k *quantile 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 quantile classes. Similarly, the *lower-closed* q_k quantile 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.

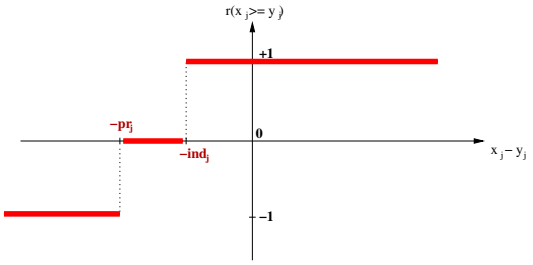
The multiple criteria extension of the single criterion q-sorting works as follows. Let $F = \{1, ..., m\}$ be a finite and coherent family of m performance criteria [3] and let x and y be two evaluation vectors on F. For each criterion j in F, we suppose the performances to be measured on a real scale $[0; M_j]$, supporting an indifference discrimination threshold ind_j and a preference discrimination threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$. 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 [5][4], every criterion $j \in F$ characterises on X a marginal double threshold order $\geqslant_j$ with the following semantics (see Fig. 1):

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

$r(x \geqslant_j y) = +1$ 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

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



significance w_j of his marginal "*at least as good as*" characterisation $r(x \geqslant_j y)$ to the global characterisation $r(x \geqslant y)$:

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

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* $(x \lll y)$ 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* $(x \ggg y)$ is observed on a concordant criterion [5].

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

$$r(x \succsim y) = \begin{cases} 0, & \text{if } [\exists j \in F : r(x \lll_j y)] \wedge \\ & \quad [\exists k \in F : r(x \ggg_k y)] \\ [\, r(x \geqslant y) \veebar -r(x \lll y) \,]^5 & \text{, otherwise.} \end{cases} \tag{3}$$

And 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 $r(x \ggg y) = 1$, and $r(x \succsim y) = -1$ if $r(x \geqslant y) \leqslant 0$ and $r(x \lll y) = 1$.

Property 1. *The bipolar characteristic of x belonging to* upper-closed *quantile class q^k, resp.* lower-closed *quantile class q_k, may be assessed as follows:*

$$r(x \in q^k) = \min [\, -r(\mathbf{q}(p_{k-1}) \succsim x),\ r(\mathbf{q}(p_k) \succsim x) \,]$$
$$r(x \in q_k) = \min [\, r(x \succsim \mathbf{q}(p_{k-1})),\ -r(x \succsim \mathbf{q}(p_k)) \,]$$

[3] Coherent means here: universal, minimal and preferentially consistent wrt marginal preferences (see [4]).

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

[5] $\veebar$ denotes the symmetric or epistemic disjunction operator. If ϕ and φ denote two propositions, $r(\phi \veebar \Phi)$ 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.

Proof : In our bipolar characteristic calculus [1, 2, 5], *logical negation* is implemented by changing the sign of the r values whereas *logical conjunction* is implemented via the min operator. The bipolar outranking relation $\succsim$, being *weakly complete*, verifies actually the *coduality principle* (see Bisdorff 2013 [5]). The dual ($\not\preceq$) of $\succsim$ is, hence, identical to the strict converse outranking $\precnsim$ relation. ■

Property 1 gives us 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 $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
For each object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$

1. $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathbf{q}(p_{k-1}) \succsim x), r(\mathbf{q}(p_k) \succsim x)\big)$
2. if $r(x \in q^k) \geqslant 0$ **add** x to q-tiles class 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 quantile classes (q). As $\mathcal{Q}$ represents a partition of the criterion performance measurement scales, there is a potential for run time optimisation.

Example 1. *We may compute a didactical example with the help of the Digraph3 Python software.* [6]

```
>>> from bigOutrankingDigraphs import *
>>> t = RandomPerformanceTableau(numberOfActons=50,seed=5)
>>> qs = QuantilesSortingDigraph(t,limitingQuantiles=5)
>>> qs.showSorting()
*--- Sorting results in descending order ---*
]0.8 - 1.0]: ['a16','a22','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]:   []
```

As made evident with this example, useful formal properties of the q-tiles sorting result 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 first and second quintile class. ($0.8 - 1.0$ and $0.6 - 0.8$). In the limit, a not yet evaluated alternative would appear sorted in all the five quintile classes.
2. *Uniqueness*: If no indeterminate outranking situations are being observed ($r() \neq 0$), each decision alternative would be sorted into exactly one single quantile class. This for instance the case with alternative 'a24' which is solely sorted into the first quintile ($0.8 - 1.0$).
3. *Independence*: The sorting result for alternative x, is independent of the other alternatives' sorting result. The independence property gives us access to efficient *parallel processing* of class membership characteristics $r(x \in q^k)$ for all $x \in X$ and q^k in $\mathcal{Q}$. Similarly, the sorting content of a quantile class is independent of the other classes' contents.

For each alternative x in X, we may thus compute a *lower* and an *upper* q-tiles sorting limit. The lower limit is determined by the one of it lowest q-tiles class whereas the upper limit is determined by the one from its highest q-tiles class. Alternative 'a24' is thus sorted into the performance class $0.8 - 1.0$ whereas 'a16' is sorted into the performance class $0.6 - 1.0$.

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

The q-tiles sorting result leaves us hence with a partition of the set of decision alternatives into more or less refined performance classes.

Example 2. *Reconsidering Example 1, we may observe, for instance, seven such performance classes:*

```
>>> from bigOutrankingDigraphs import *
>>> t = RandomPerformanceTableau(numberOfalternatives=50)
>>> qs = QuantilesSortingDigrap(t,limitingQuantiles=5)
>>> qs.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']
```

These performance classes may be linearly rank 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 class limits;
2. *Pessimistic*: In decreasing lexicographic order of, first, the lower and, secondly, the upper quantile class limits;
3. *Average*: In decreasing numeric order of the average of the lower and upper quantiles limits.

Practical experiments have shown that the '*average*' ranking strategy gives the most convincing ranking when indeterminate outranking situations and/or missing data is given.

In view of the resulting 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 outranking relation*, denoted $\gtrapprox$:

$$r(x \gtrapprox y) = \begin{cases} +1, & \text{if } \big[x \in c_i \wedge y \in c_j \wedge i < j\big] \\ -1, & \text{if } \big[x \in c_i \wedge y \in c_j \wedge i > j\big] \\ r(x \succsim y) & \text{, otherwise.} \end{cases} \tag{4}$$

$r(x \gtrapprox y) = r(x \succsim y)$ only when x and y do appear in the same performance class, i.e. $i = j$. In the limit, when, one the one hand, only one single performance class is given, we recover the standard outranking relation. On the other hand, when n singleton performance classes are given, we directly obtain a linear ranking of the decision alternatives. The corresponding decomposed digraph, denoted $G(X, \gtrapprox)$, is called a *pre-ranked sparse* outranking digraph.

Usually, depending on the number q of quantiles used in the q-tiles sorting step, a more or less refined pre-ranking is obtained.

Example 3. *Reconsider our previous Example 1 concerning a set of 50 decision alternatives. We show below a map of the standard*

[6] R. Bisdorff (2016). Tutorials of the Digraph3 resources. *Documentation of the Digraph3 Python resources*, FSTC/ILIAS Decision Systems Group, University of Luxembourg. http://leopold-loewenheim.uni.lu/docDigraph3/

outranking relation $\succsim$ (see Fig. 2. left side) and a map of the 5-tiled average limits pre-ranked outranking relation $\gtrapprox$ (see Fig. 2. right side): The 5-tiled pre-ranking of the outranking relation of Exam-

Figure 2. Map of standard $\succsim$ and pre-ranked $\gtrapprox$ outranking relation[7]

ple 1 shows seven performance classes with a minimal cardinality of 1 (c_1) and a maximal cardinality of 14 (c_5). The actual fill rate of the pre-ranked sparse $\gtrapprox$ outranking relation is 18%, i.e. the actual remaining part of the standard outranking relation $\succsim$.

The ordinal correlation [8] *between the standard ($\succsim$) and the pre-ranked ($\gtrapprox$) outranking relation, denoted $\tau(\succsim, \gtrapprox)$ 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 the set of decision alternatives. We still need to locally rank each performance class content.

2.1 Heuristic linear closures of pairwise outranking situations

To linearly rank the complete set X of decision alternatives, we need to locally rank without ties the k performance classes $c_i \subseteq X$ for $i = 1, ..., k$. To do so, we will need the pairwise valued outranking situation characteristics, i.e. $r(xSy)$ 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.

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

[8] 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.

The q-tiles ranking algorithm

1. **Input**: the outranking digraph $\mathcal{G}(X, \gtrapprox)$, 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 $\mathcal{G}_{|c_i}$
   (if ties, render alphabetic order of concerned action keys)

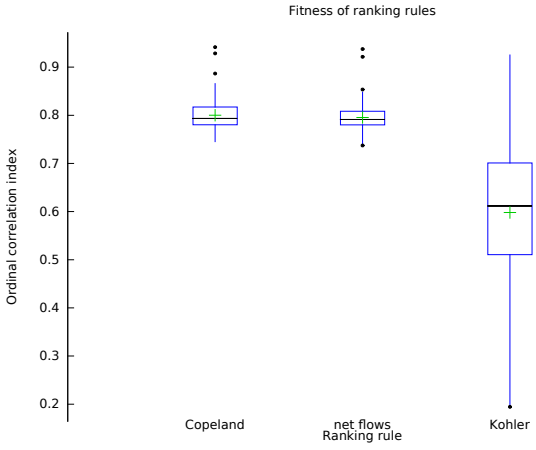
   else: $R_k \leftarrow c_i$

   append R_k to $\mathcal{R}$
3. **Output**: $\mathcal{R}$

The complexity of the q-tiles ranking algorithm is *linear* in the number k of components resulting from a q-tiles sorting. Now, component wise outranking sub-relations do only exceptionally give local linear rankings. Usually, pairwise majority comparisons do not even render complete or, at least, transitive partial relations (see Bouyssou and Pirlot 2005 [7]).

Three heuristic ranking rules are here most suitable for our purpose – *Copeland's, Net-flows' and Kohler's rule* – all three of complexity $\mathcal{O}(\#(q^k)^2)$ on each restricted outranking digraph $\mathcal{G}_{|c_i}$. In case of local *ties* (very similar evaluations for instance), the local ranking procedure will render these alternatives in increasing *alphabetic ordering* of the action keys. Only the two first of these ranking rules are in fact computationally scalable for big outranking digraphs. In Fig. 3, we may furthermore notice, that the quality of the

Figure 3. Sample of 100 random outranking graphs of order 250



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 around 0.8 [6]) when compared with Kohler's rule (median correlation 0.6 only). As Copeland's ranking rule is the simplest to implement, we will by default use this ranking rule in the sequel.

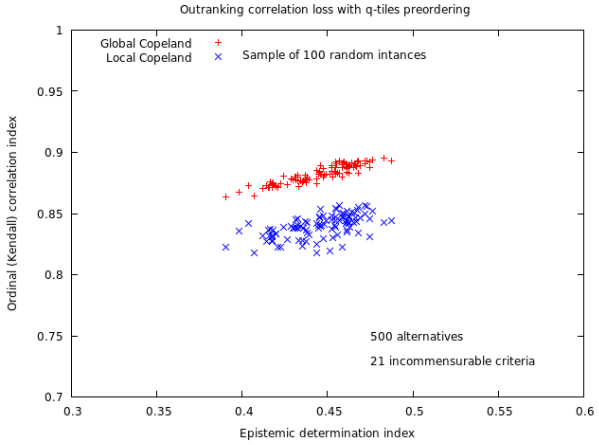
2.2 Fitness of the linear ranking result

The fitness of our pre-ranked sparse versus the standard outranking digraph model for constructing linear rankings may be appreciated when comparing both the ordinal quality [6] of the linear ranking

result obtained from a standard and from a 50-tiled pre-ranked outranking relation with the help of a sample of 100 random performance tableau gathering the performance evaluations of 500 decision alternatives on 21 incommensurable performance criteria along three equi-significant objectives (see Fig. 4).

The 50-tiles pre-ranking produces on average around 38 (sd. 7.6) performance classes, with a minimum of 32 and a maximum of 67 components, leading to a diagonal outranking fill rate of around 3% (sd: 0.4%). We furthermore notice that the ordinal correlation between the standard outranking relation and the linear ranking result obtained with the Copeland rule is around 0.88 (sd: 0.007) when the ranking is constructed from the standard and, around 0.83 (sd: 0.009), for the one constructed from the 50-tiled pre-ranked sparse outranking relation (see Fig. 4). The pre-ranking step does apparently not deteriorate significantly the quality of the ranking result. Notice furthermore that the quality of the ranking result is, both times, augmenting with the epistemic determination[9] of the standard outranking relation. It is also evident that the quality of the linear ranking result is actually depending on the performance evaluations. In the limit, when all alternatives appear to be very similarly evaluated, all linear rankings will necessarily appear badly correlated with the given outranking relation. On the other hand, if the given performance tableau shows a clear alignement of the decision alternatives, all potential ranking rules may produce more or less highly correlated linear ranking results.

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



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 independence properties of the pairwise outranking approach. Following indeed from the *independence property* of the q-tiles sorting of each alternative into each performance class, the q-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 actual ranking procedures being all local to each diagonal component c_i, these procedures may as well be safely processed in *parallel threads* on each restricted outranking digraph $\mathcal{G}_{|c}$.

Along these ideas, we have specially adapted our Digraph3 computing ressources[10] in order be able to run the linear ranking computations on the HPC clusters [9] of the University of Luxembourg[11]. All our computations are operated on the *gaia-80* node, a Delta D88x-M8-BI, 8 * Intel Xeon E7-8880 v2 @ 2.5 GHz machine 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 and our Digraph3 collection of Python3 modules [8]. We use, for instance, the following generic algorithmic design for implementing multi-threaded 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 outranking relation ($\succsim$) and, from a pre-ranked sparse one ($\gtrapprox$).

Table 1. Comparing the standard and pre-ranked sparse approach[12]

digraph order	$\succsim$ relation t_g	τ_g	$\gtrapprox$ relation t_{bg}	τ_{bg}
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 of the linear ranking result appears to be independent of the actual order, namely .88, resp. 0.83, for the standard, resp. the pre-ranked outranking relation.

These excellent run times encouraged us to furthermore develop specific cythonized[13] C versions of our Digraph3 Python modules in order to tackle pre-ranked outranking digraphs with sizes up to

[9] See Bisdorff 2012 [6].

[10] See the documentation of the Digraph3 resources FSTC/ILIAS Decision Systems Group, University of Luxembourg. http://leopold-loewenheim.uni.lu/docDigraph3

[11] S. Varrette, P. Bouvry, H. Cartiaux and F. Georgatos. 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), July 2014,IEEE, pp 959–967.

[12] Legend: – t_g, resp. t_{bg}, are the corresponding constructor run times; – τ_g, resp. τ_{bg} give the corresponding ordinal correlations between the ranking result and the given standard $\succsim$ relation.

[13] Cython: C-extensions for Python

five trillions (5×10^{12}) of pairwise outranking situations (see Table 2). For the biggest instances, we generate a random 3 decision ob-

Table 2. HPC performance measurements for big data

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

jectives performance tableau with 13 incommensurable criteria and 5% missing data.[14] For 1 000 000 alternatives (input data size = 1.4 GB) we thus obtain, 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 0.049%. 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 $1h17'$.

Finally, we may estimate the quality of the linear ranking, denoted $>_q$, of such a big outranking digraph by sampling the ranking quality on sub-digraphs of order 1 000 for instance. Here we recover in fact, by the LLN, an average sampled correlation result we have already noticed in Table 1, namely $\overline{\tau}(\succsim, >_q) \rightsquigarrow 0.83$ with a standard deviation diminishing with the number of samples we take into account. This result confirms again that the quality of our ranking approach does in fact not depend on the actual order and size of the pre-ranked sparse outranking digraph.

Conclusion

We present in this paper a pre-ranked sparse outranking digraph model coupled with a linearly ordering algorithm based on quantiles-sorting & local-ranking procedures. Global ranking results with this spare implementation of an outranking digraph fit well with those given by a standard outranking digraph.

Furthermore, independent sorting and local ranking procedures offer effective multiprocessing capacities. Efficient *scalability* allows, hence, the linear ranking of very big performance tableaux gathering several millions of evaluations.

Good perspectives for further optimisation with cython C implementations and HPC ad hoc tunings are given. All Python and cython HPC modules are freely available for further developments on the git repository of the Digraph3 ressources: 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.

REFERENCES

[1] 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

[2] R. Bisdorff (2002). Logical Foundation of Multicriteria Preference Aggregation. In: Bouyssou D et al (eds) Essay in Aiding Decisions with Multiple Criteria. Kluwer Academic Publishers 379-403

[3] 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

[4] B. Roy and D. Bouyssou (1993). Aide Multicritère à la Décision : Méthodes et Cas. Economica Paris

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

[6] 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 91–100

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

[8] R. Bisdorff (2016). Tutorials of the Digraph3 resources. Documentation of the Digraph3 resources FSTC/ILIAS Decision Systems Group, University of Luxembourg. http://leopold-loewenheim.uni.lu/docDigraph3

[9] 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 959–967

[14] R. Bisdorff. Generating performance tableaux. See Tutorials of the Digraph3 ressources.