Efficient, robust and effective rank aggregation for massive biological datasets

Pierre Andrieu, Bryan Brancotte, Laurent Bulteau, Sarah Cohen-Boulakia, Alain Denise, Adeline Pierrot, Stéphane Vialette

Highlights

Efficient, Robust and Effective Rank Aggregation for Massive Biological Datasets

Pierre Andrieu, Bryan Brancotte, Laurent Bulteau, Sarah Cohen-Boulakia, Alain Denise, Adeline Pierrot, Stéphane Vialette

- Efficient graph-based partitioning method for rank aggregation with ties
- Robust approach frontier-based to highlight common areas of all optimal consensus
- Effective solution for the life science community to rank genes from massive datasets
- Large-scale evaluation of new algorithms for rank aggregation with ties

Efficient, Robust and Effective Rank Aggregation for Massive Biological Datasets

Pierre Andrieu[a], Bryan Brancotte[b], Laurent Bulteau[c], Sarah Cohen-Boulakia[a], Alain Denise[a,d], Adeline Pierrot[a], Stéphane Vialette[c]

[a] *Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91405, Orsay, France.*
[b] *Institut Pasteur, Paris, France*
[c] *Université Gustave Eiffel, LIGM, CNRS, Marne-la-Vallée, France*
[d] *Université Paris-Saclay, CEA, CNRS, Institute for Integrative Biology of the Cell (I2BC), 91198, Gif-sur-Yvette, France.*

Abstract

Massive biological datasets are available in various sources. To answer a biological question (e.g., "which are the genes involved in a given disease?"), life scientists query and mine such datasets using various techniques. Each technique provides a list of results usually ranked by *importance* (e.g., a list of ranked genes). Combining the results obtained by various techniques, that is, combining ranked lists of elements into one list of elements is of paramount importance to help life scientists make the most of various results and prioritize further investigations. Rank aggregation techniques are particularly well-fitted with this context as they take in a set of rankings and provide a consensus, that is, a single ranking which is the "closest" to the input rankings. However, (i) the problem of rank aggregation is NP-hard in most cases (using an exact algorithm is currently not possible for more than a few dozens of elements) and (ii) several (possibly very different) exact solutions can be obtained. As answer to (i), many heuristics and approximation algorithms have been proposed. However, heuristics cannot guarantee how far from an exact solution the consensus ranking will be, and the approximation ratio of approximation algorithms dedicated to the problem is fairly high (not less than 3/2). No solution has yet been proposed to help true-users dealing with the problem encountered in point (ii).

In this paper we present a complete system able to perform rank aggregation of massive biological datasets. Our solution is *efficient* as it is based on an original partitioning method making it possible to compute a high-quality

consensus using an exact algorithm in a large number of cases. Our solution is *robust* as it clearly identifies for the user groups of elements whose relative order is the same in any optimal solution. These features provide answers to points (i) and (ii) and lie in mathematical bases offering guarantees on the computed result. Also, our solution is *effective* as it has been implemented into a real tool, ConquR-BioV2 is used for the life science community, and evaluated at large-scale using a very large number of datasets.

Keywords:
Rank aggregation, Consensus ranking, Massive biological datasets, Kemeny rule

---

1. Introduction

Huge amounts of biological data are daily reported in a large number of public databases. Such data can be mined and queried using a variety of approaches. While the approach developed in this paper can be applied to various contexts where ranked lists of elements have to be combined, our motivating examples will be based on biological sources querying as we have been working in this concrete setting so far.

Querying biological datasets can be performed using portals as provided by NCBI[1] [1], allowing users to submit a keyword to the portal to then collect a set of answers. Answers are usually provided as Web pages describing data items, and they are ranked by *relevance*, that is, by the number of occurrences of the keyword in each answer. However, properly querying such portals remains a difficult task since various formulations of the same query can be considered. Among other strategies, life scientists make use of synonymous terms (*breast cancer* versus *carcinoma of the breast*), alternative spellings (*tumour* versus *tumor*, *ADHD* versus *Attention deficit hyperactivity disorder*), or describe the concepts involved in their keywords at various levels of granularity (*Lynch syndrome* versus *colorectal cancer*). As a consequence, life scientists have to deal with several lists of hundreds of answers that need to be combined into one list to prioritize further investigations.

Rank aggregation techniques are particularly well-fitted with this context as they take in a set of rankings, that is, a set of ranked elements and provide a *consensus ranking*, that is, a single ranking that is the "closest" to the input

[1] http://www.ncbi.nlm.nih.gov/Entrez

rankings. Rank aggregation have been used in various contexts including aggregating answers returned by several Web engines [2], determining the winner in a sport competition [3], determining the winner of an election [4] or gathering answers from several biological queries [5].

However, the problem of rank aggregation is well-known to be NP-hard in most cases [2, 6, 7]. Computing an optimal consensus ranking is currently not possible for more than a few dozens of elements. As a consequence, a plethora of heuristics and approximation algorithms have been designed [2, 8, 9, 10, 11, 12, 13]. Unfortunately heuristics do not provide any guarantee with respect to the quality of the returned result and the approximation algorithms have an approximation ratio actually high (not less than 3/2) thus are not better-quality in practice than heuristics.

Moreover, many different optimal consensus rankings may exist for a same set of rankings. It turns out that the positions of some elements can vary a lot from one optimal consensus to another while the positions of other elements may be robust among the set of all the optimal consensus. Identifying the groups of elements whose relative order is the same in all the optimal solution is very informative and helpful for a user who wants to know how sound the provided consensus is. This task is though highly challenging.

Additionally, while much research has focused on the case where rankings are permutations (*i.e.*, total orders) of the same underlying set, real life applications are facing rankings with ties (elements ranked at the same position and thus placed in the same bucket). A few works considered the problem of rankings with ties: for example, [9] and [11] introduced approximation algorithms to the problem while [14] compared of most of the major heuristics and algorithms adapted to ranking with ties.

We are thus facing three challenges to make the most of rank aggregation in real settings. The first one is to develop procedures able to quickly compute optimal consensus rankings in real-life applications. The second one is to provide a way to evaluate the robustness of positions of elements in an optimal consensus. The third one is to consider solutions for rankings with ties and able to take into account missing elements.

The purpose of this paper is to tackle these challenges. First, we develop an *efficient* solution based on a process able to quickly compute consensus rankings by decomposing large datasets into much smaller ones where high quality (possibly exact) algorithms can be run. Second, we provide an algorithm able to draw frontiers between groups of elements such that their relative order in the whole set of optimal consensus rankings are inviolable,

thus giving information on the *robustness* of the positions of elements in the produced consensus ranking. Our approach lies in mathematical bases providing guarantees on the results. Third, we make our approach *effective* by providing the ConQuR-BioV2 tool where our methods are fully implemented and available to the life science community. Our approach has been evaluated at large-scale using in a very large number of massive datasets, involving rankings with ties with possibly missing elements.

The remainder of this paper is organized as follows. Section 2 provides the definitions of the major concepts underlying rank aggregation. Section 3 introduces a graph representation of the input rankings that will be used by our approach. Section 4 introduces the algorithm ParCons, an efficient partitioning procedure for rank aggregation while section 5 provides ParFront, an algorithm able to give information on the robustness of the positions of elements in the consensus ranking produced by ParCons. Section 6 presents ConQuR-BioV2, the tool we make available to the life science community where our approach has been fully implemented. Section 7 evaluates our approach on a very large number of massive biological datasets. Finally, Section 8 places the contribution in the landscape of rank aggregation approaches and draws conclusions.

2. Preliminaries

In this section we formally introduce the concept of rankings, the rank aggregation problem, and we present a running example reused all along this paper to illustrate our approach.

2.1. Definitions

Rankings. Given a set of elements U (*e.g.*, genes), a *ranking on* U is an ordered list of pairwise disjoint subsets of U called *buckets*. A ranking is *complete* if the union of its buckets is equal to U. For example, if $U = \{A, B, C, D\}$, then $r_1 = [\{A\}, \{C\}, \{B\}, \{D\}]$, $r_2 = [\{B, A\}, \{C\}, \{D\}]$ and $r_3 = [\{B, A, C, D\}]$ are three complete rankings on U, but the ranking $[\{B, A\}, \{D\}]$ is not a complete ranking on U as C is missing. Finally, $[\{B, A\}, \{C, A, D\}]$ is not a ranking as A is present twice.

Vocabulary. We set that x and y are *tied* in a ranking r if x and y are in the same bucket in r, that is, they are equally important in r. We also set that x is *before* y in a ranking r if the bucket containing x is strictly before the bucket containing y in r.

If b elements are before x in r, then the *position of x in r* is $b+1$. Note that tied elements share the same position.

Unification process. Rankings in real datasets are usually not all on the exact same set of elements. They are said to be incomplete. To complete a set of rankings, the *unification process* can be applied [15, 12] by appending at the end of each incomplete ranking r a *unification bucket*, noted $\{...\}_u$ containing all the missing elements of the ranking r. This process is used when the missing elements in rankings are considered as less important than the present elements (which is a reasonable consideration in our use case).
Illustration. Consider the set of rankings $R = \{r, s, t\}$ with $r = [\{A\}, \{C, D\}]$, $s = [\{B\}, \{C\}]$ and $t = [\{A\}, \{B, C\}, \{D\}]$. We observe that B is missing in r and both A and D are missing in s. Then, the new set of rankings after the unification process is $R_u = [r', s', t']$ with $r' = [\{A\}, \{C, D\}, \{B\}_u]$, $s' = [\{B\}, \{C\}, \{A, D\}_u]$ and $t^{'} = [\{A\}, \{B, C\}, \{D\}] = t$ as t is already a complete ranking.

Distance between two rankings. We denote r and s two complete rankings on U (possibly after a unification process).

As the unification process may induce bias by creating arbitrary ties, a pseudo-distance has been presented in [5]. This pseudo-distance is well-fitted with our biological context and will be used in this paper. Given $p \in [0, 1]$, the *Kemeny pseudo-distance*, denoted K^p, is defined as follows:
$K^p(r, s) = \sum_{\{x,y|x,y\in U\}} \overline{K^p}_{\{x,y\}}(r, s)$ where $\overline{K^p}_{\{x,y\}}(r, s) =$

- 1 if x is before y in one ranking whereas y is before x in the other one.
- p if x and y are not tied in one ranking but x and y are tied in a bucket which is not the unification bucket in the other one.
- 0 otherwise.

Note that the unification bucket has a specific treatment: there is no cost for the operation of untying elements which have been tied in the unification bucket.

Without loss of generality, for the further examples and experiments of this paper, we set $p = 1$. For the sake of readability, we set $K(r, s) = K^1(r, s)$.

Illustration. $K(s', t') = 1_{\{A,B\}}+1_{\{A,C\}}+0_{\{A,D\}}+1_{\{B,C\}}+0_{\{B,D\}}+0_{\{C,D\}} = 3$. Indeed, B is before A in $s^{'}$ whereas A is before B in $t^{'}$, C is before A in $s^{'}$

whereas A is before C in $t^{'}$ and B is before C in $s^{'}$ whereas B and C are tied in $t^{'}$. Note that even if A is before D in $t^{'}$ whereas A and D are tied in $s^{'}$, the cost is 0 for this pair as A and D are both in the unification bucket of $s^{'}$.

The Rank aggregation problem. Informally, *the rank aggregation problem* aims at finding a complete ranking, that is, the closest possible to the input rankings. Such a ranking is called an *optimal consensus* (also known as a *median*). More formally, given a set of complete rankings R and a complete ranking c, the *Generalized Kemeny score*, denoted $S(c, R)$, is the sum of the Kemeny pseudo-distances between c and each ranking in R. An *optimal consensus* of R is a complete ranking minimizing the Generalized Kemeny score: if $\mathscr{C}$ is the set of all the complete rankings on U, an optimal consensus of R, denoted c^*, is a complete ranking on U such that $\forall c \in \mathscr{C}$, $S(c^*, R) \leq S(c, R)$.

We denote *consensus* a complete ranking which can be an optimal consensus or not.

A pairwise representation. According to the Kemeny pseudo-distance, the cost (regarding a set of rankings R) of placing x before y in a consensus ranking is equal to the number of rankings $r \in R$ such that x is not before y in r (excluding the rankings in which x and y are both in the unification bucket). In a similar way, the cost of tying x and y is equal to the number of rankings $r \in R$ such that x and y are not tied in r.

Such costs are respectively denoted $before(x, y)$ and $tied(x, y)$ in this paper. Finally, the cost of placing x after y is equal to the cost of placing y before x. We also define $min(x, y) = min(before(x, y), before(y, x), tied(x, y))$.

For any two pairs of elements x and y, the ***pairwise cost matrix*** indicates the cost of placing in a consensus ranking x before, after or tied with y.

Note that if P is the set of all the ordered pairs of distinct elements of U, then the Generalized Kemeny score between a consensus ranking c and a set of rankings R can be computed using the *pairwise cost matrix* as follows

$$S(c, R) = \frac{1}{2} \sum_{(x,y) \in P} \overline{S}_c(x, y) \tag{1}$$

where $\overline{S}_c(x, y) =$

- $before(x, y)$ if x is before y in c.
- $before(y, x)$ if y is before x in c.
- $tied(x, y)$ if x and y are tied c.

The next subsection displays the pairwise cost matrix obtained on a given running example (Table 1).

Optimal consensus and frontiers. As mentioned earlier, an optimal consensus is not necessarily unique. As a consequence, we will introduce robustness properties on the set of optimal consensus based on the concept of *frontiers*.

Definition 1. *A frontier is an integer k such that the top-k elements are the same in all the optimal consensus.*

We now present a running example that will be used in the further sections to illustrate our approach.

2.2. Running example

In this subsection we introduce a running example that we use to illustrate the data structures introduced in this section (the pairwise cost matrix) but more interestingly to provide an intuition on the original solution (based on partitioning) we provide to compute a consensus and to compute frontiers.

Consider the example of Table 1 composed of six rankings (r_1 to r_6) of nine elements (A to I). This example is inspired by real use cases encountered where each ranking is a list of genes obtained from the NCBI Gene database using a given keyword (*e.g.*, *breast cancer*). When two genes are returned with the same score (*ex-aequo*), they are tied, that is, they are considered as equally important and placed in the same bucket. This is the case for genes D and E in the rankings r_1, r_2, r_5, r_6.

Illustration of the pairwise cost matrix. Table 2 presents the associated pairwise cost matrix. For the sake of readability, all the pairs have not been represented; the minimal cost for each line has been represented in bold. Remind that the pairwise cost matrix provides information on the cost induced by placing an element x before, after, or tied with an element y in a consensus ranking. For example, if we consider only the pair (D, E), we have $tied(D, E) = min(D, E)$ whereas $before(D, E) > min(D, E)$ and $before(E, D) > min(D, E)$. We can conclude that it is strictly cheaper to

$$r_1 := [\{D, E\}, \{F\}, \{I\}, \{A\}, \{B\}, \{C\}, \{G\}, \{H\}]$$
$$r_2 := [\{D, E\}, \{F\}, \{I\}, \{A\}, \{B\}, \{C\}, \{G\}, \{H\}]$$
$$r_3 := [\{E\}, \{D\}, \{B\}, \{C\}, \{A\}, \{F\}, \{I\}, \{G\}, \{H\}]$$
$$r_4 := [\{D\}, \{E\}, \{I\}, \{B\}, \{C\}, \{A\}, \{H\}, \{F\}, \{G\}]$$
$$r_5 := [\{I\}, \{D, E\}, \{C\}, \{A\}, \{B\}, \{H\}, \{G\}, \{F\}]$$
$$r_6 := [\{I\}, \{D, E\}, \{C\}, \{A\}, \{B\}, \{H\}, \{G\}, \{F\}]$$

Table 1: Example of input rankings.

Table 2: Pairwise cost matrix of the running example 1 (fragment).

x	y	$before(x, y)$	$before(y, x)$	$tied(x, y)$	$min(x, y)$
D	E	5	5	**2**	2
D	F	**0**	6	6	0
A	B	**2**	4	6	2
A	C	4	**2**	6	2
B	C	**2**	4	6	2
F	G	**2**	4	6	2
I	F	**3**	**3**	6	3
G	H	**3**	**3**	6	3

tie D and E in a consensus ranking, rather than having D before E or E before D. Now if we consider only the pair (I, F), we have $before(I, F) = before(F, I) = min(I, F)$ whereas $tied(I, F) > min(I, F)$. We can conclude that it is strictly cheaper to have I before F or F before I in a consensus ranking, rather than having F tied with I.

Intuition of how to compute a consensus following a partitioning approach. We now give an intuition of how a relevant consensus can be computed. Three points should be noticed. First, D and E are before A, B, C, F, G, H in each ranking and before I in a strict majority of rankings. As a consequence, a reasonable consensus ranking should then place D and E at the first two positions. Note that such information is provided with more precision by the pairwise cost matrix ($before(D, F) = min(D, F) = 0$ whereas $before(F, D) > min(D, F)$ and $tied(D, F) > min(D, F)$). Second, I is before A, B, C, G, H in a strict majority of rankings and before F in

a majority or rankings: a reasonable consensus ranking should then place I before $\{A, B, C, F, G, H\}$. Third, A, B and C are always placed before G and H. Moreover, A, B and C are placed before F in a majority of rankings. Again, a reasonable consensus ranking should place A, B, C before F, G and H.

This very first analysis allows to highlight the presence of four possible groups of genes. Let us name D, E group 1, I group 2, A, B, C group 3 and F, G, H group 4 in the following.

Determining which genes should be grouped together (that is, identifying the groups of genes to be considered) and then determining how genes should be ranked on the obtained partition are the two open questions we want to consider.

Let us now inspect such four groups at a finer grain level to consider the possible positions of genes within each group.

Group 1 (D and E). As explained above, placing D and E in the same bucket in the consensus ranking appears to be a very reliable choice.

Group 2 (I). As I is alone in its group thus ranking this gene within this group is trivial.

Group 3 (A, B and C). Let us consider pairwise relations between such genes. In a strict majority of rankings (4 versus 2), A is before B. It thus may appear natural to expect A before B in a consensus ranking. However, the following problem occurs: B is before C in a strict majority of rankings (4 versus 2) and C is before A in a strict majority of rankings (4 versus 2) but it is impossible to satisfy the three constraints A before B, B before C and C before A. The question of how to rank these three genes is thus not trivial.

Group 4 (F, G and H). In a strict majority of rankings, F is before G (4 versus 2). Once again, it may appear natural to expect F before G in a consensus ranking. Conversely, there is an "indifference relationship" for the pair $\{F, H\}$: in half of the rankings F is before H and in half of the rankings H is before F. As a consequence in this case, the relative order between F and H will somehow be arbitrary. The situation is similar for G and H. Finally, placing F before G appears to be a reliable choice, and placing H before F, between F and G or after G appear to be equivalent choices.

Finally, $[\{D, E\}, \{I\}, \{B\}, \{C\}, \{A\}, \{F\}, \{G\}, \{H\}]$ is an optimal consensus of the set of rankings presented in Table 1. The associated score is 34 ($2_{\{D,E\}} + 2_{\{D,I\}} + 2_{\{E,I\}} + 1_{\{I,A\}} + 1_{\{I,B\}} + 1_{\{I,C\}} + 3_{\{I,F\}} + 2_{\{A,F\}} + 2_{\{B,F\}} + 2_{\{C,F\}} + 4_{\{A,B\}} + 2_{\{A,C\}} + 2_{\{B,C\}} + 2_{\{F,G\}} + 3_{\{F,H\}} + 3_{\{G,H\}}$).

Intuition of how to compute frontiers. As seen above, the position of H seems to have more flexibility than the position of D and E. Actually $[\{D, E\}, \{I\}, \{B\}, \{C\}, \{A\}, \{H\}, \{F\}, \{G\}]$ is another possible optimal consensus, with (by definition) the same score of 34. There are a total of 9 optimal consensus for this running example. Highlighting common points between them is thus particularly important to provide the user with an information on the robust elements, that is, elements which positions do not vary a lot in the set of optimal consensus. Remind that a frontier is an integer k such that the same top-k elements are in all optimal consensus. As both D and E are before all the remaining elements in a strict majority of rankings, there is an obvious frontier after D and E. Some other frontiers are much more difficult to catch. For instance, there is another frontier between $\{D, E, I\}$ and $\{A, B, C, F, G, G\}$ as D, E, I are before A, B, C, F, G, H in all the optimal consensus.

This section has introduced the basis concepts of rank aggregation and provided an intuition on how partitioning can be exploited to compute a consensus and how frontier can be obtained. By essence, rank aggregation approaches consider pairs of elements (placed before, after or tied in the rankings). Representing rankings using graphs thus appears natural. The next section introduces graph representations of the input rankings while Section 4 describes the partitioning process we follow from these structures to compute a consensus.

3. Graph representation of rank aggregation with ties

In this section, we introduce a graph-based pairwise representation of the elements to rank. Then, we show that computing the strongly connected components of this graph allows to form coarse grain groups of elements that will be used to partition the initial problem into smaller sub-problems and to place frontiers between groups of elements.

3.1. Graph-based pairwise representation of the elements

Our challenge is both to (i) partition the initial problem into as many small sub-problems as possible and (ii) put frontiers between (robust) groups of elements. To this aim, we define G_e the *graph of elements* and R_e the set of the *robust arcs of* G_e.

In this representation, vertices are elements (the genes to rank) and for each pair of elements $\{x, y\}$, the arcs are related to which cost(s) in the pairwise cost matrix is (are) minimal among $before(x, y)$, $before(y, x)$ and $tied(x, y)$. More precisely, there is an arc from x to y if placing y before x in the consensus is more costly than placing x before y or tying x and y. This does not imply that placing x before y is the best choice: the equality could be better.

We want to distinguish two kinds of arcs (i) arcs (x, y) such that x and y could reasonably be tied and (ii) arcs (x, y) such that placing x before y is a priori the best choice. We call robust arc any arc of case (ii).

Let us now define formally G_e and R_e.

Definition 2 (G_e, graph of elements)**.** *Let* $G_e = (V_e, E_e, R_e)$ *be the directed graph such that:*

- $V_e = U$
- $E_e = \{(x, y) \in V_e^2 : before(y, x) > min(x, y)\}$
- $R_e = \{(x, y) \in V_e^2 : before(y, x) > min(x, y) \wedge tied(x, y) > min(x, y)\}$.

Arcs and Robust arcs in the graph of elements G_e. In G_e, there is **no** arc from y to x if and only if $before(x, y)$ is minimal. Moreover, if $before(x, y)$ is the unique minimum, then there is an arc from x to y, and this arc is robust (*i.e.* it is in R_e). Intuitively, regular arcs (*i.e.*, not robust) can be used to find an optimal consensus of the input rankings, and robust arcs give necessary conditions on **all** optimal consensus, which then allow us to place frontiers between groups of elements. Table 3 recaps for a given pair of elements $\{x, y\}$ the link between $before(x, y)$, $before(y, x)$, $tied(x, y)$, $min(x, y)$ and the (robust) arcs between x and y in G_e.

Interestingly, the definition of G_e fits with rankings with ties: if two elements x and y should be tied, then both placing x before y and placing y before x in a consensus is costly thus there is both an arc from x to y and another one from y to x.

Table 3: Arcs of G_e according to position of min(x,y) in the pairwise cost matrix.

$before(x,y)$	$before(y,x)$	$tied(x,y)$	in G_e	Robust arc
$=$ **min(x,y)**	$> min(x,y)$	$> min(x,y)$	$x \to y$	yes
$> min(x,y)$	$=$ **min(x,y)**	$> min(x,y)$	$x \leftarrow y$	yes
$> min(x,y)$	$> min(x,y)$	$=$ **min(x,y)**	$x \leftrightarrows y$	no
$=$ **min(x,y)**	$> min(x,y)$	$=$ **min(x,y)**	$x \to y$	no
$> min(x,y)$	$=$ **min(x,y)**	$=$ **min(x,y)**	$x \leftarrow y$	no
$=$ **min(x,y)**	$=$ **min(x,y)**	$> min(x,y)$	$x \quad y$	no
$=$ **min(x,y)**	$=$ **min(x,y)**	$=$ **min(x,y)**	$x \quad y$	no

Remark. In a previous work [16], we used a slightly different notion: we did not define robust arcs but we defined a second graph G_r named the robust graph of elements. Information provided by G_r and R_e are equivalent: it can be seen from Table 3 and the similar Table in [16] that G_r is the oriented graph on U whose arcs are all (x, y) such that $(y, x) \notin R_e$.

Illustration of the graph of elements in our running example. Figure 1 represents the graph of elements G_e related to the input rankings presented in Table 1).

Note that all the arcs in Figure 1 are robust except (D, E) and (E, D). The arcs between D and E indicate that they should be tied.
As $before(D, E) > min(D, E)$ (resp., $before(E, D) > min(D, E)$), then $before(D, E)$ (resp., $before(E, D)$) cannot be the unique minimum for the pair (D, E). In other words, the arc (D, E) (resp., (E, D)) is not robust.

Interest of G_e. The oriented arcs of G_e give an idea of which element we would like to put before the other. Indeed, when there is an arc from x to y but no arc from y to x, then $before(x, y) = min(x, y) \neq before(y, x)$ (see Table 3). However, we cannot follow every arc of G_e to build a consensus: when there is a cycle in G_e, it is impossible to respect every arc. To get rid of cycles, we compute the graph of the strongly connected components of G_e, introduced in the next section.

3.2. Graph of the strongly connected components

This subsection recalls a few definitions from graph theory, then explains why it is useful to compute the strongly connected components of G_e.

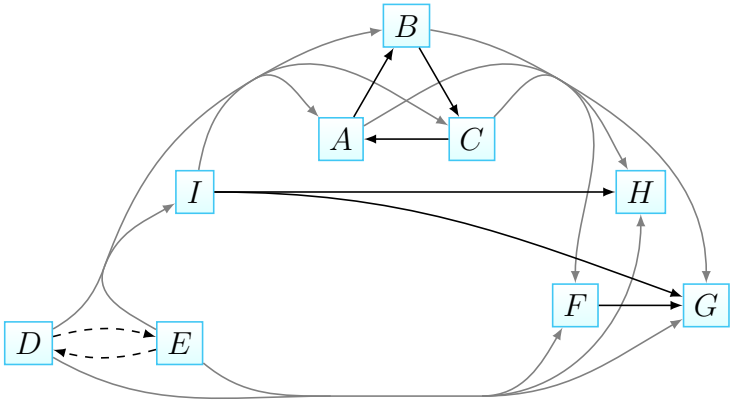

Figure 1: Graph of elements G_e for the example presented in Table 1. Solid arcs are robust, dashed arcs are not. Grey arcs with multiple sources and/or targets indicate that all pairwise arcs between the corresponding sets of vertices are present in G_e (for example, there are arcs from each of D, E to all other vertices, but not from I to F).

Recall that a strongly connected component (SCC) of a directed graph $G = (V, E)$ is a subset $V^{'}$ of V (possibly V itself) such that (i) for any two vertices (x, y) of $V^{'}$, there exists a directed path from x to y, and (ii) $V^{'}$ is maximal for (i) *i.e.* there is no subset $V^{''}$ of V such that $V^{'} \subset V^{''}$ and $V^{''}$ respects (i).

Definition 3 (G^c, graph of the strongly connected components of G_e)**.** *We denote $G^c = (V^c, E^c)$ the graph of the strongly connected components of G_e i.e. the directed graph such that*

- *V^c is the set of the strongly connected components of G_e*
- *$(c_i, c_j) \in E^c$ if and only if there is at least one element x of c_i and one element y of c_j such that (x, y) is an arc of G_e*

Computing the strongly connected components of G_e can be done with Tarjan's strongly connected components algorithm [17].

By definition, G^c is a directed acyclic graph (DAG). As a consequence, there is at least one topological sort of G^c *i.e.* a list $\mathscr{T} = [T_1, T_2, \ldots, T_k]$ of all the vertices of G^c such that T_i is not reachable from T_j for each $i < j$. A topological sort can be computed with Kahn's algorithm [18].

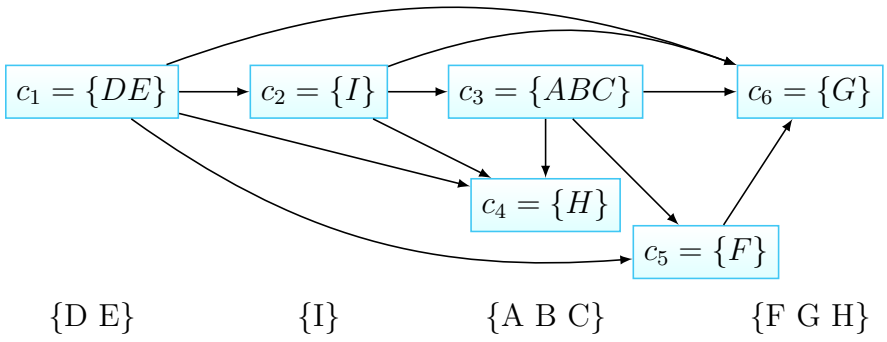

Figure 2: Top: G^c, the graph of the strongly connected components of G_e. Below: the frontiers.

Illustration. Figure 2 represents the graph G^c corresponding to our running example provided in Table 1. Computing the SCCs of G_e enables the decomposition of the universe into 6 sets: $\{D, E\}$, which should not be separated as they are tied in a strict majority of rankings, $\{I\}$, $\{A, B, C\}$ which form an incompatibility difficult to solve, and $\{F\}$, $\{G\}$ and $\{H\}$.

Interest of G^c. The graph G^c allows to partition the initial problem into as many smaller problems as possible. Indeed, vertices of G^c are sets of elements of U. Since G^c has no cycles, there is a total order on the vertices of G^c which respect the arcs. This allows to rank these sets of elements with a minimal cost (since arcs of G_e are built according to minimal costs). To compute a consensus, it will only remain to rank elements inside the vertices of G^c. This is proved in the next section.

4. Efficient rank aggregation with ties

This section introduces ParCons, a new rank aggregation procedure to efficiently provide a consensus ranking. This process is based on a fundamental property on the graph G^c presented here-after.

4.1. Fundamental property of G^c

The property of G^c presented in this section can be exploited to partition the initial problem into several sub-problems allowing to compute a high-quality consensus ranking in a reasonable time.

Let us first set two points of vocabulary. First, a consensus ranking c *respects* an ordered partition $\mathscr{T} = [T_1, T_2, ..., T_k]$ of U if and only if for all pairs of elements $\{x, y\}$ such that $x \in T_i$, $y \in T_j$ and $i < j$ then x is before y in c. Second, we denote $R(T_i)$ the set of rankings built from the input set of rankings R by removing all the elements which are not in T_i.
Illustration. Consider R the set of rankings presented in Table 1 and consider c_1 the SCC of G_e which contains D and E. Then, we obtain that $R(c_1) = R(\{D, E\}) = [r'_1, r'_2, r'_3, r'_4, r'_5, r'_6]$ where $r'_1 := [\{D, E\}]$,
$r'_2 := [\{D, E\}]$, $r'_3 := [\{E\}, \{D\}]$, $r'_4 := [\{D\}, \{E\}]$, $r'_5 := [\{D, E\}]$,
$r'_6 := [\{D, E\}]$.

Property 1 (Concatenation property)**.** *Let $\mathscr{T} = [T_1, T_2, \ldots, T_k]$ be a topological sort of G^c and μ_i be an optimal consensus for $R(T_i)$ for $1 \leq i \leq k$. Then the concatenation $\mu_1.\mu_2 \ldots \mu_k$ is an optimal consensus for R.*

Proof. Let μ be the concatenation $\mu_1.\mu_2 \ldots \mu_k$ and c be any consensus ranking. We prove that μ is an optimal consensus by showing that $S(\mu, R) \leq S(c, R)$. The score S is defined in (Equation 1 in Section 2) as a sum over $(x, y) \in P$. We cut this sum in $k+1$ parts and show that each part is smaller or equal for μ than for c: for each i from 1 to k, we consider the part of the sum over (x, y) such that x and y both belong to T_i. Then this part is smaller or equal for μ than for c since μ_i is an optimal consensus for $R(T_i)$. Now consider the remaining part of the sum. It is over (x, y) such that x and y do not belong to the same T_i. Assume that $x \in T_i$, $y \in T_j$, $i < j$ (the proof is similar if $i > j$). As T_i is before T_j in $\mathscr{T}$, there is no arc from y to x in G_e. By construction of G_e, we can conclude that $before(x, y) = min(x, y)$. In other words, the cost induced by (x, y) in μ cannot be higher than the cost induced by (x, y) in c. Finally, $S(\mu, R) \leq S(c, R)$ as claimed. □

Corollary 1. *For any topological sort $\mathscr{T} = [T_1, ..., T_k]$ of G^c, there exists an optimal consensus ranking which respects $\mathscr{T}$.*

Remark. *The concatenation property proves that there are some optimal consensus which respect the topological sorts, but there may exist some optimal consensus that do not respect any topological sort. An example is given below.*

Let us inspect the rankings presented in table 4. There are two SCC ($\{A, B, C\}$ and $\{D, E, F\}$) as no element in $\{D, E, F\}$ can reach an element in $\{A, B, C\}$. Furthermore, $[\{A\}, \{B\}, \{C\}]$ is an optimal consensus

$r_1 := [\{A\}, \{B\}, \{D\}, \{C\}, \{E\}, \{F\}]$
$r_2 := [\{A\}, \{B\}, \{D\}, \{C\}, \{E\}, \{F\}]$
$r_3 := [\{B\}, \{F\}, \{D\}, \{C\}, \{A\}, \{E\}]$
$r_4 := [\{B\}, \{C\}, \{A\}, \{F\}, \{D\}, \{E\}]$
$r_5 := [\{C\}, \{A\}, \{B\}, \{E\}, \{F\}, \{D\}]$
$r_6 := [\{C\}, \{A\}, \{B\}, \{E\}, \{F\}, \{D\}]$

$[\{A\}, \{B\}, \{D\}, \{C\}, \{E\}, \{F\}]$

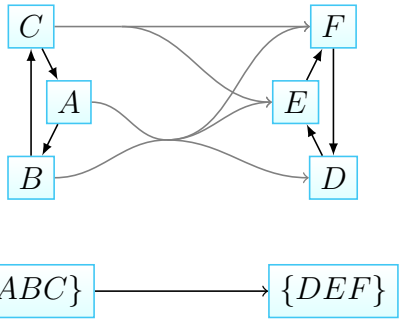

Table 4: On the left side, example of input rankings (above) and an optimal consensus (below). On the right side, the graph G_e and its graph of components G^c (all arcs between any of A, B, C and D, E, F are present and robust in G_e, *except* (C, D)). Note that the given consensus does not respect the (unique) topological sort of G^c, since D is before C.

for the sub-problem induced by $\{A, B, C\}$ *and* $[\{D\}, \{E\}, \{F\}]$ *is an optimal consensus for the sub-problem induced by* $\{D, E, F\}$*. As* C *which is the last element of* $[\{A\}, \{B\}, \{C\}]$ *has no arc to* D *which is the first element of* $[\{D\}, \{E\}, \{F\}]$*, then* C *and* D *can be switched in the optimal consensus* $[\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}]$*, thus* $[\{A\}, \{B\}, \{D\}, \{C\}, \{E\}, \{F\}]$ *is also an optimal consensus but do not respect any topological sort.*

A direct consequence of the concatenation property is that the different sub-problems $R(T_1)$, $R(T_2)$, ..., $R(T_k)$ can be treated independently. This observation naturally leads to the algorithm ParCons we present now.

4.2. A partitioning algorithm for rank aggregation

Thanks to the *concatenation property*, we know that having an optimal consensus for each sub-problem obtained with G^c guarantees an optimal consensus for the initial problem.

The question is now how to treat the elements within each SCC of G_e. Two cases can be considered. If (case 1), for any pair of elements $\{x, y\}$ in the SCC, $tied(x, y) = min(x, y)$ then one optimal consensus for the sub-problem is a single bucket containing all the elements of the SCC. Otherwise (case 2), no trivial optimal consensus can be found: an auxiliary ranking algorithm (depending on the size of SCC, an exact algorithm or rather a heuristic or an approximation algorithm) should be called to provide a consensus ranking for the sub-problem induced by the SCC.

Algorithm 1 (ParCons) describes the procedure providing a consensus ranking. Note that the result of the returned consensus ranking may depend

on the auxiliary algorithms used to solve the sub-problems. Moreover, if we never call a heuristic or an approximation algorithm (either if case 1 always holds or if an exact algorithm is used instead), then the consensus provided is necessarily optimal.

Algorithm 1: ParCons: Graph-based procedure providing a consensus ranking (with a pre-process and an auxiliary algorithm).

```
Input: R : set of rankings
begin
    M ← PairwiseCostMatrix(R)
    G_e ← ComputeGraphGe(M)
    G^c ← SCC(G_e)
    topolSortGe ← TopologicalSorting(G^c)
    nbSccGe ← numberSCCofGe
    consensus ← EmptyListOfSets()
    for i ← 1 to nbSccGe do
        if ∃v1 ≠ v2 in topolSortGe[i] with tied(v1, v2) > min(v1, v2)
         then
            subProblem ← CopyInput(R)
            Remove from subProblem all the elements not in
             topolSortGe[i]
            subConsensus ← auxiliaryAlgo(subProblem)
            Append each bucket of subConsensus to consensus
        else
            Append topolSortGe[i] to consensus
Result: consensus
```

Illustration. Let us inspect again the example presented in Table 1. A possible topological sort for G^c is: $[c_1, c_2, c_3, c_5, c_6, c_4]$ *i.e.*

$[\{D, E\}, \{I\}, \{A, B, C\}, \{F\}, \{G\}, \{H\}]$. Given this ordering, it is now possible to quickly construct several parts of the final consensus ranking of R. Indeed, we know that: (i) $[\{D, E\}]$, (ii) $\{I\}$, (iii) $[\{F\}]$, (iv) $[\{G\}]$ and (v) $[\{H\}]$ are optimal consensus rankings for the sub-problem induced respectively by (i) c_1, (ii) c_2, (iii) c_5, (iv) c_6 and (v) c_4 (case 1).

Although placing A, B and C is not trivial (case 2), we can already claim that there exists an optimal consensus in which (i) D and E share the 1^{st} position ex-aequo ; (ii) I is placed at the 3^{rd} position; (iii) A, B and C share

the 4^{rd}, 5^{th} and 6^{th} position (still not ordered yet); (iv) F is in $7th$ position; (v) G is in 8^{th} position; (vi) H is in 9^{th} position.

Suppose now that the algorithm used to solve the sub-problem induced by c_3 placed A before both B and C and placed B before C. Then, the obtained consensus ranking for the example presented in Section 1 is $[\{D, E\}, \{I\}, \{A\}, \{B\}, \{C\}, \{G\}, \{F\}, \{H\}]$. Continuing with our running example, we now provide an intuition of the reasons why there is no trivial optimal consensus in case 2. Let us notice that there is at least a pair (x, y) of elements in the SCC such that the transitivity is not respected regarding the minimal costs of each pair of the triple. For example, in c_3 of Figure 2, we have a path from A to B and a path from B to A. As a consequence, we would like to place A before B or tied with B, and B before A or tied with A. The only way to respect these constraints is to tie them. However, $tied(A, B) > min(A, B)$. As a consequence, we have no guarantee that A and B are tied in an optimal consensus. Computing an optimal consensus is here intrinsically difficult.

5. Frontiers for Robust Rank Aggregation

While Section 4 was dedicated to the problem of efficiently computing a consensus, this section focuses on frontiers, used to identify robust areas in the returned consensus. More precisely, this section first introduces the ParFront algorithm and the mathematical properties it is based on, and second it shows that ParFront is able to compute strictly more frontiers than any other approach of the related work.

5.1. Computing frontiers

Frontiers provide an answer to the problem of multiple (different) optimal consensus for a given set of input rankings by highlighting sets of elements whose relative order is the same in all the optimal consensus. Frontiers thus allow to make the user aware of the areas of the consensus which are robust and areas which may vary a lot from an optimal consensus to another one.

More precisely this subsection introduces ParFront (Algorithm 2) which computes an ordered partition $P = [P_1, P_2, ..., P_k]$ of the set of elements to rank, such that all the optimal consensus respects P. ParFront is based on Theorem 1 which provides sufficient conditions for an ordered partition to be respected by all the optimal consensus.

Theorem 1. *Let $G_e = (V_e, E_e, R_e)$ be the graph of elements. Let $P = [P_1, P_2, ..., P_k]$ be an ordered partition of V_e such that*

1. $\forall i < j$, $\forall x \in P_i$, $\forall y \in P_j$, $(y, x) \notin E_e$, *and*

2. $\forall i$, $\forall x \in P_i$, $\forall y \in P_{i+1}$, $(x, y) \in R_e$.

Then each optimal consensus respects P.

Proof. Consider a consensus ranking c which does not respect P. We will show that c is not optimal. From c, we build a consensus c' as follows: take first the elements of P_1 in the same order as they are in c, then append the elements of P_2 in the same order as they are in c, and repeat the operation for the elements of P_3, ..., until P_k. By construction, c' respects P, thus c' is different from c. Now, let us compare $S(c, R)$ and $S(c', R)$ using Equation 1 of Section 2. Each pair of elements (x, y) such that x and y are in the same relative order in c and c' induces the same cost for c and c'.

Now consider pairs (x, y) such that x and y are not in the same relative order in c and c' (there is at least one such pair). By construction of c', x and y are in different groups of P. Suppose $x \in P_i$ and $y \in P_j$ with $i < j$. Then by hypothesis on P, there is no arc from y to x in G_e, *i.e.* $before(x, y) = min(x, y)$. Hence, $\overline{S}_{c'}(x, y) \leq \overline{S}_c(x, y)$.

Moreover, since $y \in P_j$ is before $x \in P_i$ in c, there exists $i \leq k < j$ and $z \in P_k$, $z' \in P_{k+1}$ such that z' is before z in c. But by construction of c', z is before z' in c'. By the theorem hypotheses, $(z, z') \in R_e$, *i.e.* $before(z, z')$ is the unique minimum, hence $\overline{S}_{c'}(z, z') < \overline{S}_c(z, z')$.

Finally, $\overline{S}_{c'}(x, y) \leq \overline{S}_c(x, y)$ for every pair (x, y) and there is a strict inequality for at least one pair, so overall $S(c, R) > S(c', R)$, hence c is not optimal, concluding the proof. □

Recall that we defined a *frontier* as an integer k such that the set of the k first elements is the same in all the optimal consensus. An ordered partition $P = [P_1, P_2, ..., P_\ell]$ satisfying the conditions of Theorem 1 allows to draw $\ell - 1$ frontiers (between each consecutive parts of the partition). The goal is then to find a partition with as many parts as possible, satisfying the conditions of Theorem 1. Note that it is not possible to find a partition both satisfying these conditions and having more parts than the number of strongly connected components of G_e. Indeed, item 1. of Theorem 1 is satisfied if and only if the partition P is *coarser* than a topological sort of G^c (i.e. each part of P is the union of strongly connected components of G_e, ordered according to a topological sort). To find a partition with as many parts as possible satisfying the conditions of Theorem 1, we start with a topological sort of G^c

(thus satisfying condition 1), and repeatedly merge consecutive parts that do not satisfy condition 2. This is done by the algorithm ParFront (Algorithm 2 below).

Algorithm 2: ParFront: returns frontiers according to Theorem 1.

Input: $T = [T_1, ..., T_k]$: list of SCC of Ge according to a topological sort of G^c , Re: set of robust arcs of G_e

begin

```
/* Part 1:  compute the ordered partition P                    */

P ← T /* P starts as a copy of T: from now on P
   satisfies condition 1                                        */
i ← 1
while i < size(P) do
    /* check condition 2 between P[i] and P[i+1]               */
    if ∃x ∈ P[i], y ∈ P[i+1], (x, y) ∉ Re then
        P[i] ← P[i] ∪ P[i+1] /* merge P[i] and P[i+1]          */
        delete P[i+1]
        i ← max(i−1, 1) /* backtrack one step                  */
    else
        i ← i+1 /* move forward one step                       */

/* Part 2:  use P to compute the frontiers                     */

frontiers ← emptyList()
frontier ← 0
for i ← 1 to size(P) do
    frontier ← frontier + size(P[i])
    add frontier in frontiers
```

Result: List of integers (positions of the frontiers)

Remark. *Algorithm ParFront can be used in two different ways. First, the user may want to get a coarse-grained ranking of the elements, then ParFront can be run alone to provide a robust coarse-grained ranking (the partition P). Alternatively, the user may want to get a "reliability-aware" total ranking of*

the elements, then ParFront is run after Algorithm ParCons to provide a consensus with robust frontiers.

In the following, we prove that the result of Algorithm 2 is the best possible following Theorem 1, *i.e.* it is the unique partition which has the maximal number of parts among those satisfying the hypotheses of the Theorem.

At first, we recall that an ordered partition $P = [P_1, \ldots, P_k]$ is a *refinement* of another $P' = [P'_1, \ldots, P'_\ell]$ if any part of P' is a union of consecutive parts of P. Informally, this means that P is a further fragmentation of P'.

Property 2. *The partition given by Algorithm 2 satisfies the hypotheses of Theorem 1. Moreover, this partition is a refinement of all other partitions which satisfy these hypotheses.*

Note that this property implies that the result of Algorithm 2 is independent of the topological sort given as input (indeed if P refines P' and P' refines P, then $P = P'$).

Proof. We say that an ordered partition P is a *frontier-partition* if it satisfies the conditions of Theorem 1.

The proof lies in two parts: first we prove that Algorithm 2 yields a frontier-partition, then we state that it is a refinement of any frontier-partition.

For the first part, we first prove condition 1: at any point during the algorithm, there is no arc between $P[j]$ and $P[i]$, with $j < i$. Indeed, it is true at the initialisation of P (since we start with a topological sort), and is not invalidated by merging consecutive parts. Moreover, during the while loop, condition 2 is satisfied for each pair $P[j], P[j+1]$, where j is smaller than the current value of i. As we leave the loop when $i = size(P)$, condition 2 is then satisfied for every pair $P[i]$, $P[i+1]$, so Algorithm 2 yields a frontier-partition.

For the second part, first note that at any point of the algorithm, there cannot be any path from a vertex of $P[j]$ to a vertex of $P[i]$, with $i < j$ (this is a consequence of condition 1). We call this property the *path property*. We now consider a frontier-partition P', and show by induction that at any point of the algorithm, P is a refinement of P'. Indeed, P starts as T and T is a refinement of P' (otherwise P' would violate condition 1). Now for the induction step, we prove that each time P is modified, it remains a refinement of P'. If P is modified, it means that there is some i and some x, y such that $x \in P[i], y \in P[i+1], \{x, y\} \notin R_e$. Since P is a refinement of P', there

are some j, j' such that $P[i] \subseteq P'[j]$ and $P[i+1] \subseteq P'[j']$. Our goal is to show that $j = j'$: then when merging $P[i]$ and $P[i+1]$, the result is still a refinement of P'. Aiming at a contradiction, assume that $j \neq j'$. We have $j < j'$ (otherwise, since P' satisfies condition 2, there would be a path from y to x, which is excluded from the path property). If $j' = j + 1$, then P' violates condition 2 (indeed we assumed $\{x, y\} \notin R_e$). So $j + 1 < j'$. We consider $z \in P'[j+1]$: there exists a robust path (actually an arc) from x to z and a robust path from z to y, so from the path property, z cannot be in a part strictly before x in P (i.e. not before $P[i]$), nor in a part strictly after y (not after $P[i+1]$). So z is in either $P[i]$ or $P[i+1]$, together with either x or y. This contradicts the induction hypothesis, since at the beginning of this step P is a refinement of P' and z is not in the same part of P' as x or y. We obtained the desired contradiction, showing that when merging $P[i]$ and $P[i+1]$, the result is still a refinement of P'. Overall, P remains a refinement of P' throughout the algorithm.

Overall, the algorithm outputs a frontier-partition that is a refinement of every other frontier-partition: thus it gives a maximum-size frontier-partition, which is actually unique.

□

5.2. Comparison with frontiers of the related work

Although we are not aware of any work having introduced means to help users being aware of robust areas among the variety of optimal solutions, previous works, notably [3, 19] have introduced algorithms to "strongly" partition the elements to be ranked and have thus defined concepts that can be rethought in terms of frontiers.

In this Section, we show that the frontiers computed by ParFront are the most general in the sense that it finds strictly more frontiers than all the frontiers-like defined in previous works.

5.2.1. Non-dirty candidates

Betzler et al. [3] define a "non-dirty candidate (for 0.75)" as an element x such that for any element $y \neq$ x, either x is before y in at least 75% of the rankings, or y is before x in at least 75% of the rankings. Betzler et al. have proved that:

- all the elements placed before x in at least 75% of the rankings are then placed before x in all the optimal consensus.

- all the elements placed after x in at least 75% of the rankings are then placed after x in all the optimal consensus.

This theorem was obtained for permutations but we can prove that it remains true for complete rankings with ties with $p = 1$.

Remark that finding a non-dirty candidate x leads to an ordered partition $\mathscr{B} = [B_1, B_2, B_3]$ of the ranking in three parts:

- B_1 is the set of all the elements placed before x in at least 75% of the rankings
- $B_2 = \{x\}$
- B_3 is the set of all the elements placed after x in at least 75% of the rankings

Let us perform the following procedure: if there is a non-dirty candidate x, compute B_1 and B_3 and recursively search for non-dirty-candidates in B_1 and B_3 (an element in B_1 can be a non-dirty candidate in B_1 even if it was a dirty candidate in $B_1 \cup B_2 \cup B_3$).

This procedure is deterministic: the order in which we pick the non-dirty candidates does not affect the partition produced. Indeed, remark that if an element is a non-dirty candidate in $B_1 \cup B_2 \cup B_3$, then it remains a non-dirty candidate in B_1 or B_3.

We denote $\mathscr{B} = B_1, B_2, ..., B_b$ the partition obtained after the recursive procedure.

Property 3. *All the frontiers given by the partition $\mathscr{B}$ (non-dirty candidates) are also given by the partition P computed by ParFront (Algorithm 2).*

Proof. From Property 2, the partition P obtained by ParFront refines any partition satisfying the hypotheses of Theorem 1. So it sufficient to prove that $\mathscr{B}$ respects the hypotheses of Theorem 1.

First, let us prove that $\mathscr{B}$ respects condition 1 of Theorem 1. Let $i < j$, $x \in B_i$, $y \in B_j$. Let z be the non-dirty candidate which separates x and y (z may be x or y). Since $i < j$, we know that either $x = z$, or x is before z in at least 75% of the rankings. Similarly, we know that either $y = z$, or z is before y in at least 75% of the rankings. In all cases, we have x before y in at least 50% of the rankings, so $before(x, y) = min(x, y)$, that is, $(y, x) \notin E_e$. Thus $\mathscr{B}$ respects hypothesis 1 of Theorem 1.

Second, let us prove that $\mathscr{B}$ respects condition 2 of Theorem 1. Let $i < b$, $x \in B_i$, $y \in B_{i+1}$. By construction of B, either x is a non-dirty candidate with respect to $B_{i-1} \cup B_i \cup B_{i+1}$, or y is a non-dirty candidate with respect to $B_i \cup B_{i+1} \cup B_{i+2}$. In both cases, x is before y in 75% of the rankings, that is, $(x, y) \in R_e$ thus $\mathscr{B}$ respects condition 2 of Theorem 1. Finally, P refines $\mathscr{B}$.

□

5.2.2. Extended Condorcet Criterion

Truchon proves in [19] that for an odd number of complete rankings (that may have ties), all the optimal consensus respect any ordered partition $\mathscr{X} = [X_1, X_2, ..., X_k]$ such that $\forall i < j$, $\forall x \in X_i$, $\forall y \in X_j$, x is before y in a majority of rankings. This theorem is actually the so-called *Extended Condorcet Criterion*. Note that Truchon sets $p = 1$ for the penalty cost of creating/breaking ties.

We can (slightly) extend this result so that it remains true for an even number of rankings. The following condition thus holds: all the optimal consensus respects any ordered partition $\mathscr{X} = [X_1, X_2, ..., X_k]$ such that $\forall i < j$, $\forall x \in X_i$, $\forall y \in X_j$, x is before y in a strict majority of rankings.

We now consider that $\mathscr{X}$ is the maximal partition (maximal in terms of number of parts) respecting the condition above.

Property 4. *All the frontiers given by the partition $\mathscr{X}$ (Extended Condorcet Criterion) are also given by the partition P computed by ParFront (Alg. 2).*

Proof. Let us prove that $\mathscr{X}$ respects the two conditions of Theorem 1. First, let us set $i < j$, $x \in X_i$, $y \in X_j$. By hypothesis, x is before y in a strict majority of rankings. We obtain $before(x, y) < before(y, x)$ and $before(x, y) < tied(x, y)$, then $(y, x) \notin E_e$ and $\mathscr{X}$ respects hypothesis 1 of Theorem 1. Moreover, as $before(x, y)$ is the unique minimum, $(x, y) \in R_e$. Thus $\mathscr{X}$ respects hypothesis 2 of Theorem 1. Finally, P refines $\mathscr{X}$. □

5.2.3. Robust graph of elements

In a previous work [16], we defined the *robust graph* G_r to compute frontiers: $G_r = (V_r, E_r)$ is the directed graph such that

- $V_r = U$
- $E_r = \{(x, y) \in V_r^2 : x \neq y \wedge (before(y, x) \geq before(x, y) \vee before(y, x) \geq tied(x, y)\}$.

We proved that all the optimal consensus respect the unique topological sort $\mathscr{T} = [T_1, T_2, ..., T_k]$ of the graph of strongly connected components of G_r.

Property 5. *All the frontiers given by the partition $\mathscr{T}$ (strongly connected components of the robust graph) are also given by the partition P computed by ParFront (Algorithm 2).*

Proof. We prove that $\mathscr{T}$ respects the hypotheses of Theorem 1. First, let $i < j$, $x \in X_i$, $y \in X_j$. As $\mathscr{T}$ is a topological sort of the graph of strongly connected components of G_r, x and y are in different SCC of G_r and we have $(y, x) \notin E_r$. By definition of E_r, $(y, x) \notin E_r$ implies $before(x, y)$ is the unique minimum, that is $(x, y) \in R_e$. We can conclude $\mathscr{T}$ respects hypothesis 2 of Theorem 1. Moreover, since $before(x, y) = min(x, y)$, we have $(y, x) \notin E_e$ and $\mathscr{T}$ respects hypothesis 1 of Theorem 1. Finally, P refines $\mathscr{T}$. □

. We have considered related work having defined concepts that can be rethought in terms of frontiers and proved that ParFront computes all such frontiers. As far as we know, ParFront is thus currently the most general approach able to identify robust areas in the consensus.

6. ConQuR-BioV2

This section introduces ConQuR-BioV2 that we have implemented and which is concretely in-use for the life science community at the following address: `http://conqur-bio.lri.fr`.

ConQuR-Bio (initially introduced in [5]) has been designed in close collaboration with several members of the life science community. Current users come from the following institutes: the APHP (Hospitals in Paris area, `https://www.aphp.fr/`), the Institut Curie (Hospital and research center in oncology, `https://institut-curie.org/`) and the Institute for Integrative Biology of the Cell (I2BC, `https://www.i2bc.paris-saclay.fr/`).

The aim of ConQuR-Bio is to provide a list of genes associated with a disease (expressed as a keyword by users). ConQuR-Bio exploits the fact that disease names may have various synonyms, each synonym allows to find a list of genes and eventually provides users with a consensus list of genes.

While ConQuR-BioV1 has been used by several life scientists, it suffered from two major weaknesses : (i) the number of elements obtained as answer to a query (genes associated with a keyword) was constantly augmenting, there were thus a crucial need for an approach able to scale while not systematically

using heuristics; (ii) ConQuR-Bio has been used to provide new research hints that may be then confirmed by wet experiments; life scientist users have expressed their need to be aware of the robustness associated with the positions of elements.

ConQuR-BioV2 addresses these two key points by introducing two new key features: (i) the rank aggregation module has been rebuilt to use the ParCons algorithm and (ii) the user interface has been adapted to exploit and display the information on the frontiers produced by the algorithm ParFront.

After a brief presentation of the architecture of ConQuR-BioV2, this section provides an example of query showing the benefit for a life scientist of using ConQuR-BioV2.

6.1. Architecture

The architecture of ConQuR-BioV2, described in Figure 3, is composed of three main modules.

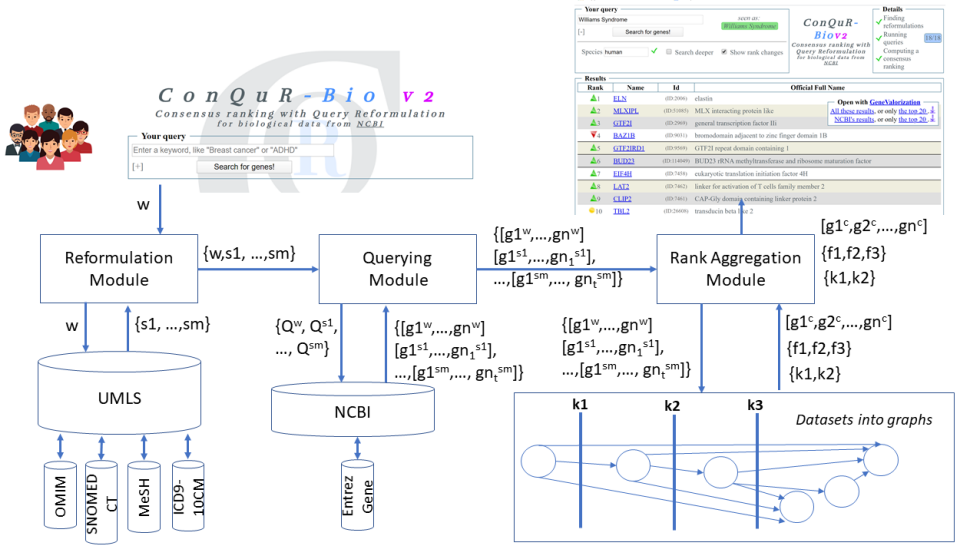

Figure 3: Architecture of ConQuR-BioV2

The first module - the Reformulation Module - takes as input w, the user keyword (*e.g.*, breast cancer) and generates a set of synonyms of w ($\{s_1, ..., s_m\}$). Such a generation is obtained by automatically using the UMLS Metathesaurus [20] which queries five databases described below and

selected with our biologist collaborators to provide synonyms for disease names.

- MeSH (Medical Subject Headings) [21] contains a large set of medical keywords as it has been designed to index the keywords associated with medical publications in PubMed.
- SNOMED CT (SNOMED Clinical Terms) [22] is a collection of medical terms providing terms, synonyms and definitions used in clinical documentation and reporting.
- The two latest versions of the International Classification of Diseases (ICD-9-CM and ICD-10-CM) [23, 24], have been developed by the World Health Organization. ICD-9-CM is the official system of assigning codes to diagnoses and procedures for hospital utilization (in the United States and Europe). ICD-10-CM contains additional codes for mortality causes.
- The Online Mendelian Inheritance in Man (OMIM) [25] contains a very rich catalog of all known genetic diseases in the human genome.

The second module - the Querying Module - takes in the user keyword and the list of synonyms provided by the reformulation module and sends one query to the NCBI EntrezGene database per keyword and synonym. For each keyword (and synonym) NCBI EntrezGene provides a list of genes associated with it. In Figure 3, $g1^w, ..., gn^w$ is the list of genes associated with the keyword w. Each list of genes is ranked by NCBI using the number of occurrences of the keyword (or the synonym) in the NCBI gene entry. As a result, the querying module outputs several ranked lists of genes (one per keyword and synonym).

The third module - the Rank Aggregation Module - takes in a set of gene lists and uses the algorithm ParCons described in Section 4 to compute a consensus ranking (denominated $g1^c...gn^c$ in Figure 3) and provide users with the frontiers (denominated $\{k1, k2, k3\}$ in Figure 3) computed using the algorithm ParFront. It is worth noticing that the Rank Aggregation module has been completely rethought from V1 to V2 making it able to manage massive datasets (using now ParCons) and to provide users with precise information on the robustness the consensus area (using ParFront).

In the example provided in Figure 4 the user has considered the keyword *Williams Syndrome* for which 18 synonyms have been found. The ordered list of genes and frontiers can be visualized by the user.

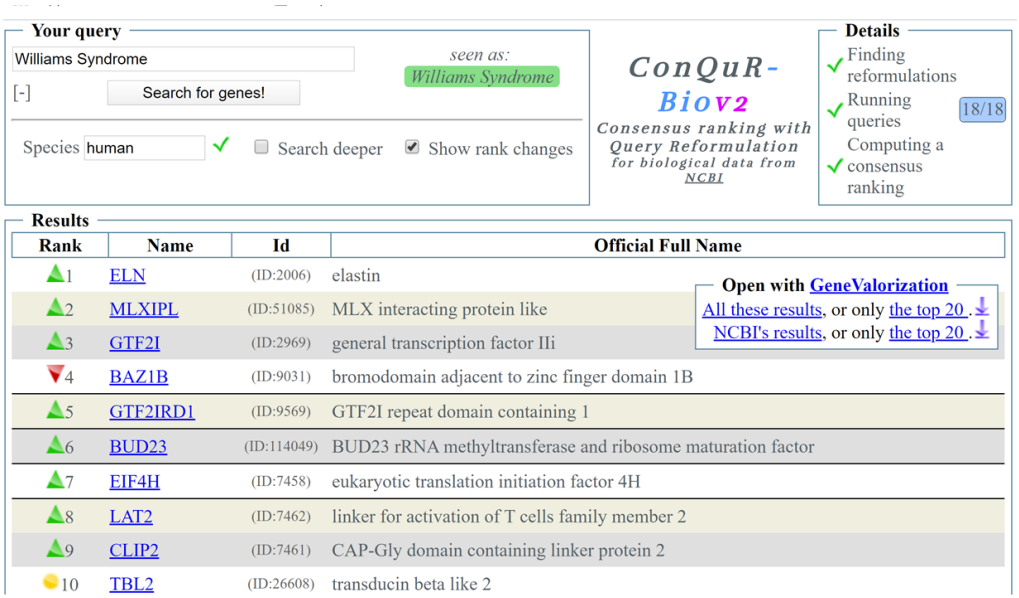

Figure 4: Interface of ConQuR-BioV2 and results obtained for the Williams Syndrome.

6.2. Interest of using ConQuR-BioV2

We consider here a real use case of ConQuR-BioV2, based on the search for genes related to the *Williams Syndrome*, a disorder having clinical manifestations of various kinds including cardiac anomalies and mental retardation. The Williams Syndrome is a rare disease (still occuring in about 1 birth out of 7,500) named very differently in the literature (*e.g.*, "Beuren Syndrome", "Hypercalcemia Supravalvar Aortic Stenosis"). According to *Orphanet* (orphanet.org), the reference source of information on rare diseases, eight genes are known to be equally strongly associated with the disease, namely (in alphabetical order), BAZ1B, CLIP2, ELN, GTF2I, GTF2IRD1, LIMK1, RCF2, and TBL2.

Using ConQuR-BioV2, the user can type *Williams Syndrome* and click on "search for genes". The result obtained is provided in Figure 4.

Note that the disease name has automatically been coloured in green as it is recognized as a MeSH term. Then, the reformulation module has run

and ,as indicated on the right side of the interface, has obtained 18 reformulations from the UMLS Metathesaurus (that is, 18 alternative names). Last, the consensus ranked list of genes is provided. The position of each gene in the original ranking of EntrezGene NCBI, that is, the ranking provided by NCBI with the user keyword (here "Williams Syndrome") is provided for information. When ConQuR-BioV2 ranked the gene higher (respectively, lower) than NCBI then a green up arrow (respectively a red down arrow) is displayed; when ConQuR-BioV2 has provided a gene that NCBI does not provide then a yellow sun is displayed.

A deeper look at the results shows two very interesting points

- the 8 genes from *Orphanet* have been retrieved by ConQuR-BioV2 while NCBI has not found TBL2;
- the top-5 (resp., top-10) genes of ConQuR-BioV2 contains 4 (resp., 6) genes from *Orphanet* while NCBI has only 2 of such genes in its top-10.

A last point to notice is the information offered by the frontiers: ConQuR-BioV2 provides users with four frontiers at position 4, 5, 6 and 7, indicating that genes should be ranked considering the top-4 elements (ELN, MLXIPL, GTF2I, BAZ1B) first then the 5th (GTF2IRD1) then the 6th (BUD23) and then 7th (EIF4H) element of the list.
The next section is dedicated to the evaluation of ConQuR-BioV2 in a very large number of real settings involving massive datasets.

7. Evaluation

The evaluation of our approach is two folds. First, we have performed a large-scale quantitative series of experiments to evaluate the ability of our approach to provide consensus on a variety of massive datasets. Second, we have performed a qualitative series of experiments to evaluate the ability of our approach to generate consensus of interest for true users.

7.1. Quantitative Evaluation

7.1.1. Experimental setting

Datasets. In this first series of experiments we have considered all the disease names from the MeSH terminology having at least 4 synonyms and associated with at least 100 genes according to NCBI EntrezGene. As a result, we have

considered 1,354 datasets of size 100 to 1,557 (295 in average) having 4 to 120 reformulations (11.5 in average).

Algorithms. Several studies of rank aggregation algorithms have been performed in the past ten years. One of the most recent is [14] which introduced an exact algorithm (reused in this paper) and concludes on the fact that three heuristics are considered as the currently most effective namely, KwikSort [10], Copeland method [26] and BioConsert [12]. We thus considered the exact algorithm and the three heuristics above that we all slightly adapted to handle the pseudo-distance described in Section 2 to handle ties and incomplete rankings properly.

Running time and quality of the results. Experiments were conducted on a four dual-core processor Intel Core 2.9GHz with 32GB memory desktop using Java 1.8.0. Each running-time measure was preceded by a warm-up time to ensure that all classes were already loaded in the JVM memory. Implementations were single-threaded.

As for the quality of the results obtained, we classically considered the *gap* [27], [15] which consists in normalizing the distance to show the additional disagreement one consensus c has compared to an optimal consensus c^*.

Given a set of rankings R and an optimal consensus c^*, the *gap* is defined as follows: $gap(c, R) = \frac{S(c,R)}{S(c^*,R)} - 1$. The value of the gap is 0 if and only if the provided consensus is an optimal consensus. If a consensus c_1 has a lower gap than another consensus c_2, it means that c_1 is better quality than c_2.

Biological datasets usually contain too many elements to compute an optimal consensus with an exact algorithm. In these cases we use as a reference the consensus ranking with the lowest score that we denote c^+(the best provided by a nonexact algorithm) and use the *m-gap* [14]: $m\text{-}gap(c, R) = \frac{S(c,R)}{S(c^+,R)} - 1$. Note that if an optimal consensus is found $m\text{-}gap(c, R) = gap(c, R)$.

7.1.2. Large-scale evaluation on massive biological datasets

We now consider five experiments performed on the 1,354 biological datasets described above. The first three are related to the evaluation of the partitioning approach while the last two focus on the robustness of the obtained consensus rankings (frontiers).

Experiment 1: Interest of the partitioning phase. The first experiment aims at evaluating the role of the partitioning in the final provided consensus. Figures 5 and 6 show that a large amount of genes can be ranked by the partitioning phase without any need of using any algorithm or heuristic. More precisely, Figure 5 shows that 100% of the genes are placed (that is, an optimal consensus is found) using only the partitioning in 396 datasets. Additionally, more than 80% of the genes are placed by the partitioning in 665 datasets (this number is obtained by considering the 100% and the 80-99% parts of Figure 5, with 665 =269+396 datasets).

Moreover, Figure 6 shows that 67.5 % (representing a total amount of 269 564 genes) of the genes from all the datasets are placed using the partitioning only.

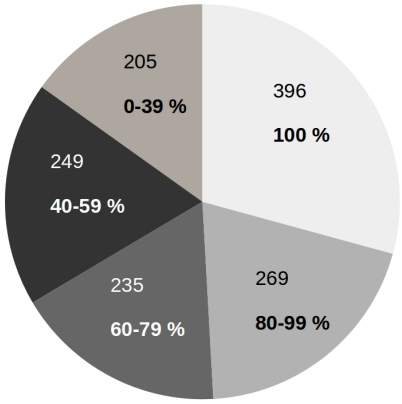

Figure 5: Number of datasets where the proportion of genes placed in the consensus by the partitioning is in a given range (0-39%, 40-59%, 60-79%, 60-99%, 100%).

To rank the remaining sets of elements, there are two possibilities. The first (and obviously preferred when possible) is to use an exact algorithm which provides best quality results but may be impossible to use if the set of elements to rank is too large. The second one (default) is to use a heuristic. The next experiment evaluates the number of cases where ParCons uses the exact algorithm.

Experiment 2: Ability to use an exact algorithm. ParCons uses the exact algorithm to compute a consensus ranking for a sub-problem if the number of genes to rank in the sub-problem is lower than 80 (*i.e.* the strongly

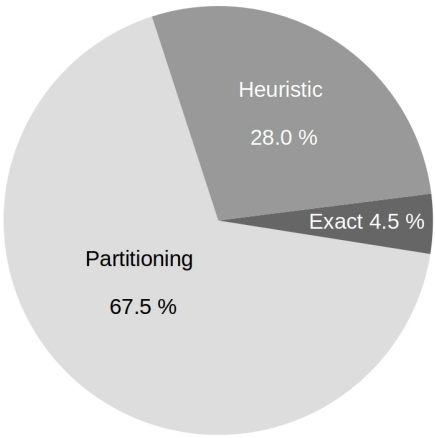

Figure 6: Total number of genes placed in the consensus ranking by the partitioning, an exact algorithm, and a heuristic.

connected component of G_e contains less than 80 genes). This ensures that the final consensus ranking can be returned to the user in a reasonable time (a few seconds).

Figure 7 provides the proportion of datasets where the consensus provided to the user has been obtained using (i) the partitioning only (cf Experiment 1), (ii) the partitioning and one exact algorithm only, (iii) the partitioning and a heuristic. Note that both (i) and (ii) provide an optimal consensus. Thanks to the exact algorithm, optimal consensus have been found in 60.4 % of the datasets (29.2 + 31.2) corresponding to 818 datasets. For information, a total of 287,531 genes (representing 72% of the total number of genes) can be positioned by the partitioning and an exact algorithm. Figure 7 indicates that a heuristic has to be used for 39.6% of the datasets which corresponds to a number of 536 datasets.

Determining the best heuristic to choose is a key point that we consider in the next experiment.

Experiment 3: Benching various heuristics with/out partitioning. In this experiment we compare the m-gap and the computation time of the three heuristics mentioned in subsection 7.1.1, used with and without partitioning, on the 1,354 datasets.

Table 5 provides the results of the bench we conducted. Two points can

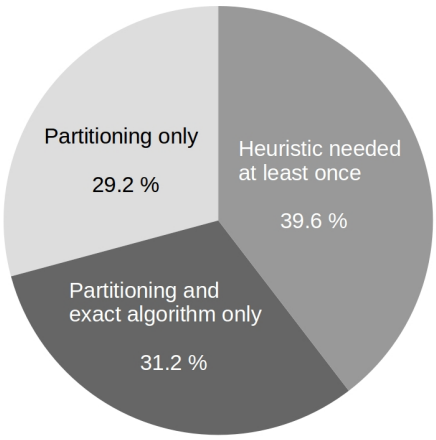

Figure 7: Proportion of datasets such that the partitioning was only used (optimal consensus), the partitioning and an exact algorithm were only used (optimal consensus), a heuristic was needed (consensus possibly non optimal).

be noticed. First, using the partitioning phase allows to increase the quality of fast algorithms while not impacting their computation time in a significant way for the user. More precisely, the m-gap (best value = 0) of KwikSort has been decreased by 47.0% and the m-gap of CopelandMethod has been decreased by 59%. Computation time remains very reasonable (less than one second for KwikSort and Copeland with the partitioning process).

Second, using the partitioning allows to increase the speed of high-quality algorithms still improving their quality. More precisely, the computation time of BioConsert has been reduced by 65% while its m-gap has been reduced by 30%.

Based on these experiments, the implementation of ConQuR-BioV2 runs systematically the exact algorithm and BioConsert in parallel. If the exact algorithm provides the result in less than 5 seconds, this result is used. Otherwise, the result of BioConsert is used. More generally, our partitioning allows us to run in parallel auxiliary algorithms on different connected components.

Experiment 4: Benefit of using frontiers. Frontiers have been designed to guide users in the confidence they may have in the provided ranking. The frontiers found at the beginning of the ranking (top first positions) are of

Heuristics	Mean m-gap	Mean time	Max time
$CopelandMethod$	$1.14 * 10^{-2}$	13 ms	695.5 ms
$ParCons$ using $CopelandMethod$	$6.74\ 10^{-3}$	60 ms	1.4 s
$KwikSort$	$1.30 * 10^{-2}$	3 ms	19.6 ms
$ParCons$ using $KwikSort$	$6.83 * 10^{-3}$	19 ms	983 ms
$BioConsert$	$4.61 * 10^{-5}$	207 ms	9.7s
$ParCons$ using $BioConsert$	$3.23 * 10^{-5}$	72 ms	2.94 s
$ParCons$ using $exact(SCC < 80)$ and $BioConsert$	$2.41 * 10^{-5}$	1.0 s	20.5 s $\leq 5s$ in 93% of datasets

Table 5: Benchmark: comparison of the quality of the provided consensus (m-gap) and the computation time. A better-quality consensus has a lower m-gap.

particular interest as most users mainly focus on the first genes of the consensus. Three points can be noticed from our experiments. First, 73% of our datasets have a frontier at position 1, that is for 73% of the diseases there is one gene which is ranked at the first position in all the optimal consensus.

The second and third points are shown in Figure 8 which provides the average number frontiers between 1 and k: our datasets have in average 10 frontiers for k=50 and the number of frontiers is higher in the top first elements. Such results are particularly interesting as they demonstrate the ability of computing frontiers useful for the user in real use cases.

Experiment 5: Ability of computing new frontiers of interest. In this second part of experiments dedicated to frontiers, we consider all the datasets and compare the number of frontiers obtained by ParFront with the number of frontiers obtained by the non-dirty elements (NDE) [3], the Extended Condorcet Criterion (ECC) [19] and the robust graph of elements (RGE) [16], as introduced in Section 5. To be fair towards all methods, we considered unified rankings: if a gene x is present in a ranking r whereas another gene y is missing in r, we considered that x is before y in r. This consideration increases the number of frontiers found using ECC and NDE methods.

Recall that Property 3 states that ParFront necessarily finds all the frontiers given by NDE, ECC and RGE methods (proved in Section 5). This

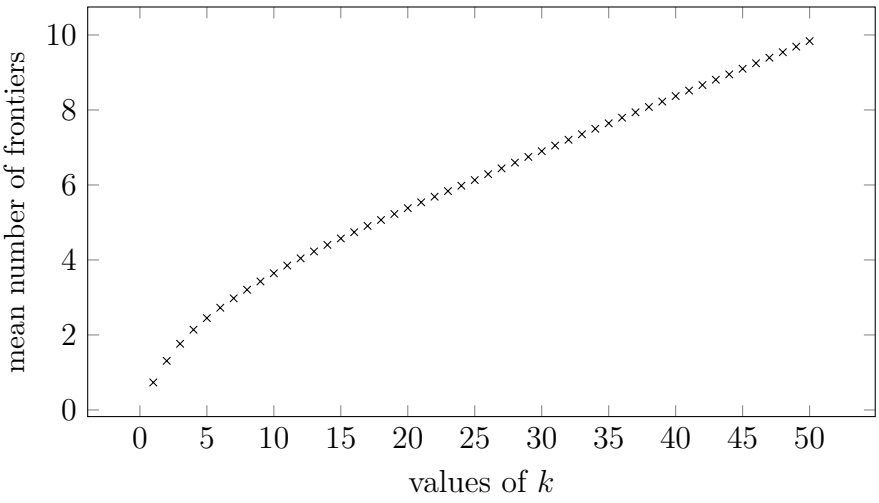

Figure 8: Mean number of frontiers between position 1 and k, k = 1 to 50 (all datasets considered).

experiment aims to quantify how much more frontiers can be obtained using ParFront. The result, displayed in Figure 9, is remarkable: ParFront (our approach) finds 1.6 times more frontiers than the RGE method, almost 15.7 times more frontiers than the ECC method and 134.7 times more frontiers than NDE method. Interestingly, Figure 9 also shows that although the NDE method provides much less frontiers than the ECC and RGE methods, NDE is able to find (and characterize) frontiers which are neither found by ECC nor found by RGE while they are all obtained by ParFront.

7.2. Qualitative study (using a gold standard)

7.2.1. Datasets

This second series of experiments aims at comparing the consensus currently provided by NCBI EntrezGene with our rank aggregation based approach. To do so, we have used a very well annotated and curated database, namely *Orphanet* [29], as a gold standard. More precisely, we selected the disease names associated with a MeSH term for which *Orphanet* was able to provide at least one gene known to be involved with the disease and available in EntrezGene. A total of 234 datasets has then been considered. Interestingly, *Orphanet* provides information on the reason why a gene is associated with a disease, allowing us to consider three categories of genes: (i) genes of category 1 are strongly associated with the disease, they correspond to ther-

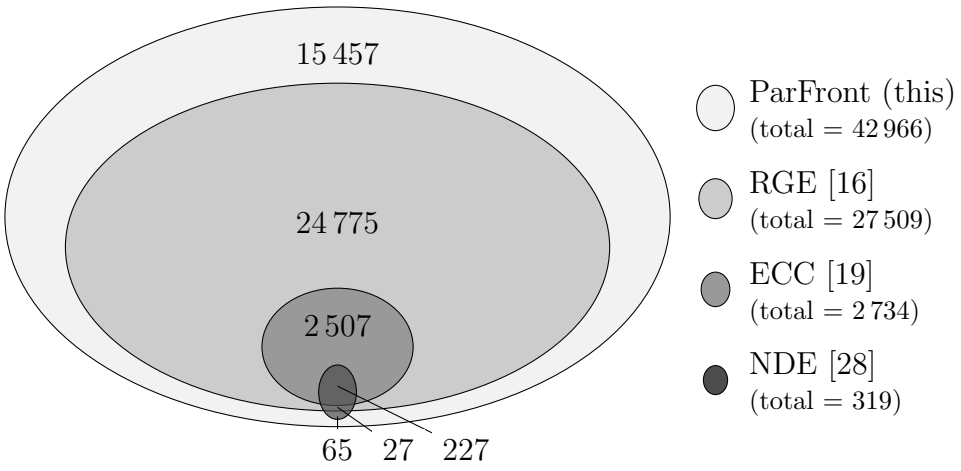

Figure 9: Venn diagram indicating the number of frontiers obtained on the dataset by each of the four methods: NDE non-dirty elements (Betzler et al. [28]), the ECC Extended Condorcet Criterion (Truchon [19]), RGE (*robust graph of elements*) and ParFront (*robust arcs*) versions.

apeutic targets or disease-causing germline or somatic mutation(s) affecting the function of the gene, (ii) genes of category 2 corresponds to genes that modify the clinical presentation of the disorder or used to monitor disorder activity, (iii) genes of category 3 are candidate genes for which a mutation associated with a disorder is suspected or genes playing a role in chromosomal rearrangements that may have a role in the disorder.

7.2.2. Results obtained by ConQuR-BioV2 compared to NCBI EntrezGene

We conducted three experiments to compare the genes belonging to the Orphanet gold standard (GS) found in the top-20 of ConQuR-BioV2 versus the top-20 of NCBI EntrezGene. We focus on the top-20 as this is the size of results provided in the first page of NCBI EntrezGene. Genes of category 1, 2 and 3 have been considered. Experiment (a) focuses on the datasets and determines whether ConQuR-BioV2 or NCBI provides more genes from the gold standard in its top-20; experiment (b) sums the number of genes from the gold standard obtained in all the top-20 of ConQuR-BioV2 and NCBI while experiment (c) inspects the position of the first gene from the gold standard found in the top-20 of the two systems.

All these experiments have also been conducted considering the top-40 genes, varying the categories of genes considered, considering a subset of the

dataset with diseases associated with at least 5 genes in the gold standard. In all such cases, results obtained were highly similar to the results presented in this section (thus not displayed).

Experiment (a): Which of ConQuR-BioV2 or NCBI provides more genes from the GS. Figure 10 shows that ConQuR-BioV2 finds a strictly higher number of genes from the gold standard in its top-20 than NCBI in 23.1% of the datasets while NCBI finds a strictly higher number of genes from the gold standard in its top-20 than ConQuR-BioV2 in only 3.8% of the datasets. Interestingly, this means that in 95.2 % of the datasets (73,08%+23,08% and corresponding to 225 datasets) ConQuR-BioV2 finds the same number or a strictly higher number of genes from the gold standard in its top-20 than NCBI.

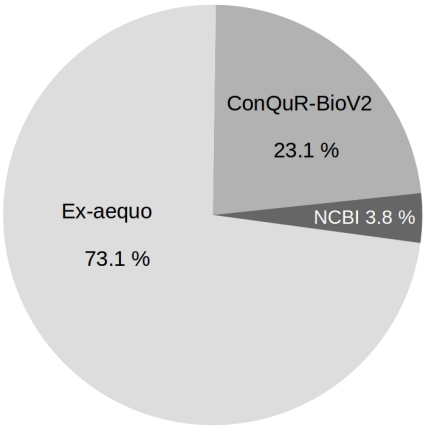

Figure 10: Number of datasets such that (i) the top-20 of ConQuR-BioV2 contains strictly more genes from the gold standard than the top-20 of NCBI (ConQur-BioV2), (ii) the top-20 of NCBI contains strictly more genes from the gold standard than the top-20 of ConQuR-BioV2 (NCBI),(iii) ConQuR-BioV2 and NCBI both have the same number of genes from the gold standard in their top-20 (Ex-aequo).

Experiment (b): Total number of genes from the GS. Figure 11 provides the total number of genes from the gold standard (summed on all the datasets) obtained in the top-20 of ConQuR-BioV2 and NCBI EntrezGene: ConQuR-BioV2 contains 1,27 times more genes from the gold standard than NCBI. Additionally, the number of genes found by ConQuR-BioV2 only is actually

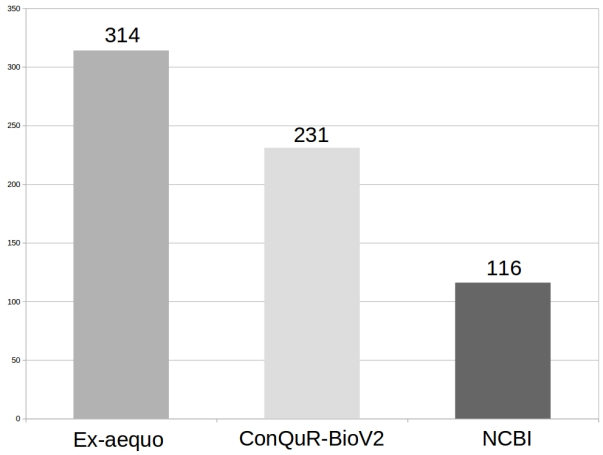

Figure 11: Sum on all the datasets of the number of genes from the gold standard (i) in the top-20 of both ConQuR-BioV2 and NCBI, (ii) in the top-20 of ConQuR-BioV2 only, (iii) in the top-20 of NCBI only.

twice more important than the number of genes found by NCBI only. Finally (not represented in the figure), note that NCBI does not find any gene of the gold standard in 44 datasets while this is only the case in 12 datasets for ConQuR-BioV2.

Experiment (c): Position of the best ranked gene from the GS. The last experiment has considered the position of the best ranked gene from the gold standard and provides information on whether such a gene is ranked before, at the same position or after in ConQuR-BioV2 compared to NCBI. Figure 12 shows that the best ranked gene from the gold standard returned by ConQuR-BioV2 is ranked strictly better than the best ranked gene from the gold standard returned by NCBI in 26.5% of the datasets. Interestingly, Figure 12 means that the best ranked gene of ConQuR-BioV2 from the GS is ranked equally or higher than the best ranked gene of NCBI from the GS in 91% of the datasets.

8. Discussion

8.1. Summaries of the contributions

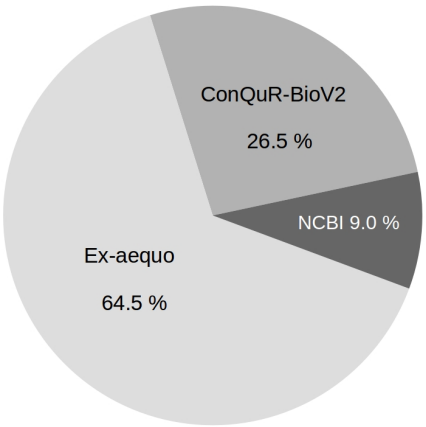

Figure 12: Number of datasets such that the position of the first gene of the GS in the top 20-consensus is (i) better than (ConQur-BioV2), (ii) worse than (NCBI) or (iii) the same as (Ex-aequo) the position of the first gene of the GS in the top 20-NCBI.

We have considered a problem encountered by most life scientists when querying biological databases: dealing with several rankings of answers to be combined into one single ranking. In this paper we provide a system based on rank aggregation techniques available to the life science community. Our first contribution consists in providing an *efficient* solution by introducing ParCons, an algorithm able to partition the very large set of elements to be ranked into smaller sets. This contribution is based on a property we demonstrated that (i) fits with rankings with ties and incomplete rankings (ii) ensures the concatenation of the solutions obtained on subsets of elements is a solution of the complete set. Results obtained on real datasets demonstrate how useful this contribution can be, allowing rank aggregation approaches to scale and provide high quality results within a reasonable time (a few seconds) for several hundreds of elements.

Our second contribution allows to provide a *robust* solution. Here we tackle the problem of multiple optimal solutions to the rank aggregation problem by exploring further the concept of *frontier* defined in [16].

Frontiers lead to highlighting sets of elements in the consensus which are robust, that is, which position does not vary among the variety of optimal solutions. In other words, any element placed between a frontier at position k_1 and a frontier at position k_2 is necessarily at a position between k_1

(excluded) and k_2 (included) in all optimal solutions. Frontiers give insight about the level of confidence the user can have on the given result. Here, we provide ParFront, a new algorithm which ensures the soundness of the identification of frontiers in datasets.

Interestingly, this new algorithm not only finds (much) more frontiers than in our previous work [16] (as shown in Figure 9, section 7), but it also subsumes the two major following previous works: the non-dirty elements [3] and the Extended Condorcet Criterion [19].

Our last (but not least) contribution relies on providing an *effective solution* by making the online tool ConQuR-BioV2, where both algorithms ParCons and ParFront have been implemented, available to the community. Our experiments demonstrate (i) the efficiency of our algorithms to provide high-quality results on real data and (ii) the relevance of using rank aggregation techniques on biological datasets for real users.

8.2. Related work

We now place our solution in the landscape of rank aggregation solutions.

General families of algorithms used to tackle the rank aggregation problem. As seen from the introduction, the rank aggregation problem has been studied in many fields: algorithmics (*e.g.*, [11]), databases (*e.g.*,[8]), social choice theory (*e.g.*, [4]), biology (*e.g.*, [30]), physics (*e.g.*, [31]), statistics (*e.g.*, [32]), artificial intelligence (*e.g.*, [33]). Works mostly focus on permutations, *i.e.* when the rankings to aggregate are complete and without ties [3, 10, 11, 27, 32, 34, 35, 7, 36]. More recently, a few works have addressed rank aggregation with ties [15, 37] and/or aggregation of incomplete rankings [38, 13, 39] have also been investigated.

Our contribution belongs to the latter category where rankings with ties are considered and missing elements are taken into consideration by using a dedicated score.

Optimality and massive datasets. Faced with the intrinsic complexity of the rank aggregation problem, many algorithms (heuristics, approximations, exact algorithms) have been developed [2, 12, 9, 10, 11, 26, 8, 35, 13, 40, 41, 3, 15], with different approaches (integer programming, dynamic programming, local search, etc.), some of them using partitioning procedures.

Exact algorithms can only handle small rankings. Other approaches can apply to very large datasets, but none of them is able to ensure optimality.

By contrast, our method allows the user to know the provided consensus being optimal in many cases (61% of our 1354 datasets), even for a large number of genes (100 to 1577).

Use of graphs for the rank aggregation problem. Graph methods have naturally been used in the context of rank aggregation. Many results on complexity (including the NP-hardness) have actually been proved using a reduction to the well-known feedback arc set problem [2, 6, 42, 43, 44]. The feedback arc set problem has also been used to design exact algorithms, approximation algorithms and/or heuristics [3, 6, 35, 45, 36]. These works mainly focus on permutations. We have developed a more general setting for dealing with incomplete rankings with ties, leading to efficient and effective algorithms.

Dealing with many optimal consensus: frontiers. Very few papers tackle the problem of multiple optimal consensus. Both [19] and [31] suggest to compute all the optimal consensus and provide a single consensus ranking using the mean position of each element in all the optimal consensus. Unfortunately this kind of solution cannot make sense on datasets with many elements as the computation time would dramatically increase. To face this problem, we have defined frontiers that highlight common points between all the optimal solutions. We proved that the frontiers produced by the ParFront algorithm include all the frontiers which can be defined using the non-dirty elements [3], the Extended Condorcet Criterion [19] and the robust graph of elements [16]. Moreover, our experiments show that ParFront draws much more frontiers than [3, 19, 16] can do.

Therefore, ConQuR-BioV2 allows the user to be aware of the robustness of the displayed consensus, better than any other approach.

8.3. Perspectives

Ongoing work consists in collecting the feedback of ConQuR-BioV2 users. We also plan to extend our approach to other kinds of biological datasets: (i) considering various kinds of datasets from NCBI (including proteins and SNPs), (ii) considering gene rankings obtained by various omics experiments (proteomics, micro-arrays). Such new contexts of study may reveal new requirements in particular concerning the distance to consider or the heuristics to favor.

From a purely theoretical point-of-view, there may be some cases where Theorem 1 is not able to provide all frontiers. We currently work on this

point, hoping to find more general sufficient conditions, or even necessary and sufficient conditions.

References

[1] D. Maglott, J. Ostell, K. D. Pruitt, T. Tatusova, Entrez gene: gene-centered information at NCBI, Nucleic acids research 33 (2005) D54–D58.

[2] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: Proc. of the WWW Conference, WWW '01, ACM, New York, NY, USA, 2001, pp. 613–622.

[3] N. Betzler, R. Bredereck, R. Niedermeier, Theoretical and empirical evaluation of data reduction for exact Kemeny rank aggregation, Autonomous Agents and Multi-Agent Systems (2013) 1–28.

[4] K. Arrow, A. Sen, K. Suzumura, Handbook of Social Choice and Welfare, volume 1, Elsevier, 2002.

[5] B. Brancotte, B. Rance, A. Denise, S. Cohen-Boulakia, ConQuR-Bio: Consensus ranking with query reformulation for biological data, in: Data Integration in the Life Sciences, Springer, 2014, pp. 128–142.

[6] T. Biedl, F. J. Brandenburg, X. Deng, On the complexity of crossings in permutations, Discrete Math. 309 (2009) 1813–1823.

[7] G. Bachmeier, F. Brandt, C. Geist, P. Harrenstein, K. Kardel, D. Peters, H. G. Seedig, k-majority digraphs and the hardness of voting with a constant number of voters, Journal of Computer and System Sciences 105 (2019) 130 – 157.

[8] R. Fagin, R. Kumar, D. Sivakumar, Efficient similarity search and classification via rank aggregation, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM, 2003, pp. 301–312.

[9] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee, Comparing and aggregating rankings with ties, in: Proc. of PODS, ACM, 2004, pp. 47–58.

[10] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: Ranking and clustering, J. ACM 55 (2008) 23:1–23:27.

[11] N. Ailon, Aggregation of partial rankings, p-ratings and top-m lists, Algorithmica 57 (2010) 284–300.

[12] S. Cohen-Boulakia, A. Denise, S. Hamel, Using medians to generate consensus rankings for biological data, in: Scientific and Statistical Database Management, volume 6809, 2011, pp. 73–90.

[13] J. A. Aledo, J. A. Gámez, D. Molina, Approaching the rank aggregation problem by local search-based metaheuristics, Journal of Computational and Applied Mathematics 354 (2019) 445 – 456.

[14] B. Brancotte, B. Yang, G. Blin, S. Cohen Boulakia, A. Denise, S. Hamel, Rank aggregation with ties: Experiments and analysis, Proc. of the VLDB Endowment (PVLDB) 8 (2015) 2051.

[15] F. Schalekamp, A. van Zuylen, Rank aggregation: Together we're strong, in: Proc of ALENEX, 2009, pp. 38–51.

[16] P. Andrieu, B. Brancotte, L. Bulteau, S. Cohen-Boulakia, A. Denise, A. Pierrot, S. Vialette, Reliability-aware and graph-based approach for rank aggregation of biological data, in: 2019 15th International Conference on eScience (eScience), IEEE, San Diego, France, 2019, pp. 136–145.

[17] R. Tarjan, Depth first search and linear graph algorithms, SIAM Journal on Computing 1 (1972).

[18] A. B. Kahn, Topological sorting of large networks, Commun. ACM 5 (1962) 558–562.

[19] M. Truchon, An extension of the Condorcet criterion and Kemeny orders, Technical Report, Centre de Recherche en Économie et Finance Appliquées (CREFA), Université Laval, Quebec, Canada, 1998.

[20] O. Bodenreider, The Unified Medical Language System (UMLS): integrating biomedical terminology, Nucleic Acids Research 32 (2004) D267–D270.

[21] C. E. Lipscomb, Medical subject headings (mesh), Bulletin of the Medical Library Association 88 (2000) 265–271.

[22] M. Stearns, C. Price, K. Spackman, A. Wang, Snomed clinical terms: Overview of the development process and project status, Proceedings / AMIA ... Annual Symposium. AMIA Symposium (2001) 662–6.

[23] W. H. Organization, International classification of diseases : [9th] ninth revision, basic tabulation list with alphabetic index, 1978.

[24] W. H. Organization, Icd-10 : international statistical classification of diseases and related health problems : tenth revision, 2004.

[25] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, V. A. McKusick, Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders, Nucleic Acids Research 33 (2005) D514–D517.

[26] A. H. Copeland, A reasonable social welfare function, 1951. Seminar on Appl. of Mathematics to the social sciences, University of Michigan.

[27] A. Ali, M. Meilă, Experiments with Kemeny ranking: What works when?, Mathematical Social Sciences 64 (2012) 28 – 40. Computational Foundations of Social Choice.

[28] N. Betzler, J. Guo, C. Komusiewicz, R. Niedermeier, Average parameterization and partial kernelization for computing medians, Journal of Computer and System Sciences 77 (2011) 774 – 789. JCSS IEEE AINA 2009.

[29] S. Weinreich, R. Mangon, J. Sikkens, M. E. e. Teeuw, M. Cornel, [orphanet: a european database for rare diseases], Nederlands tijdschrift voor geneeskunde 152 (2008) 518—519.

[30] R. Kolde, S. Laur, P. Adler, J. Vilo, Robust rank aggregation for gene list integration and meta-analysis, Bioinformatics (Oxford, England) 28 (2012) 573–80.

[31] S. Muravyov, I. Marinushkina, Intransitivity in multiple solutions of Kemeny ranking problem, Journal of Physics Conference Series 459 (2013) 012006.

[32] A. Rajkumar, S. Agarwal, A statistical convergence perspective of algorithms for rank aggregation from pairwise data, in: E. P. Xing, T. Jebara (Eds.), Proceedings of the 31st International Conference on Machine Learning, volume 32 of *Proceedings of Machine Learning Research*, PMLR, Bejing, China, 2014, pp. 118–126.

[33] A. Korba, S. Clemencon, E. Sibony, A Learning Theory of Ranking Aggregation, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, PMLR, Fort Lauderdale, FL, USA, 2017, pp. 1001–1010.

[34] J. P. Barthelemy, A. Guenoche, O. Hudry, Median linear orders: Heuristics and a branch and bound algorithm, European Journal of Operational Research 42 (1989) 313–325.

[35] A. Davenport, J. Kalagnanam, A computational study of the Kemeny rule for preference aggregation, in: Proceedings of the 19th National Conference on Artifical Intelligence, AAAI'04, AAAI Press, 2004, p. 697–702.

[36] C. Kenyon-Mathieu, W. Schudy, How to rank with few errors, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, NY, USA, 2007, p. 95–103.

[37] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee, Comparing partial rankings, SIAM J. Discret. Math. 20 (2006) 628–648.

[38] S. Amodio, A. D'Ambrosio, R. Siciliano, Accurate algorithms for identifying the median ranking when dealing with weak and partial rankings under the Kemeny axiomatic approach, European Journal of Operational Research 249 (2016) 667 – 676.

[39] E. Moreno-Centeno, A. R. Escobedo, Axiomatic aggregation of incomplete rankings, IIE Transactions 48 (2016) 475–488.

[40] J. C. de Borda, Mémoire sur les élections au scrutin, Histoire de l'académie royale des sciences, 1781, pp. 657–664.

[41] I. Azzini, G. Munda, A new approach for identifying the Kemeny median ranking, European Journal of Operational Research 281 (2020) 388 – 401.

[42] G. Even, J. S. Naor, B. Schieber, M. Sudan, Approximating minimum feedback sets and multi-cuts in directed graphs, in: Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization, Springer-Verlag, Berlin, Heidelberg, 1995, p. 14–28.

[43] J. Bartholdi, C. A. Tovey, M. A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare 6 (1989) 157–165.

[44] M. R. Garey, D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., USA, 1990.

[45] V. Conitzer, A. Davenport, J. Kalagnanam, Improved bounds for computing Kemeny rankings, in: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06, AAAI Press, 2006, p. 620–626.