Reliability-Aware and Graph-Based Approach for Rank Aggregation of Biological Data

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

Reliability-aware and graph-based approach for rank aggregation of biological data

Pierre Andrieu
LRI, CNRS, U. Paris-Sud
U. Paris-Saclay, France
pierre.andrieu@u-psud.fr

Bryan Brancotte
Institut Pasteur
Paris, France
bryan.brancotte@pasteur.fr

Laurent Bulteau
LIGM, CNRS, U. Paris-Est
Marne-la-vallée, France
laurent.bulteau@u-pem.fr

Sarah Cohen-Boulakia
LRI, CNRS, U. Paris-Sud
U. Paris-Saclay, France
sarah.cohen-boulakia@u-psud.fr

Alain Denise
LRI & I2BC, CNRS, U. Paris-Sud
U. Paris-Saclay, France
alain.denise@u-psud.fr

Adeline Pierrot
LRI, CNRS, U. Paris-Sud
U. Paris-Saclay, France
adeline.pierrot@u-psud.fr

Stéphane Vialette
LIGM, CNRS, U. Paris-Est
Marne-la-vallée, France
vialette@univ-mlv.fr

***Abstract*—Massive biological datasets are available in public databases and can be queried using portals with keyword queries. Ranked lists of answers are obtained by users. However, properly querying such portals remains difficult since various formulations of the same query can be considered (e.g., using synonyms). Consequently, users have to manually combine several lists of hundreds of answers into one list. Rank aggregation techniques are particularly well-fitted to this context as they take in a set of ranked elements (rankings) and provide a consensus, that is, a single ranking which is the "closest" to the input rankings. However, 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. A plethora of heuristics have thus been proposed which behaviour are, by essence, difficult to anticipate: given a set of input rankings, one cannot guarantee how far from an exact solution the consensus ranking provided by an heuristic will be. The two challenges we want to tackle in this paper are the following: (i) providing an approach based on a pre-process to decompose large data sets into smaller ones where high-quality algorithms can be run and (ii) providing information to users on the robustness of the positions of elements in the consensus ranking produced. Our approach not only lies in mathematical bases, offering guarantees on the result computed but it has also been implemented in a real system available to life science community and tested on various real use cases.**

***Index Terms*—rank aggregation, graph approach, consensus ranking, biological datasets**

I. Introduction

Huge amounts of biological data are daily reported in a large number of public databases. Such data can be queried 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 describes 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 to this context as they take in a (multi)set of ranked elements (rankings) and provide a *consensus ranking*, that is, a single ranking that is the "closest" to the input 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. On the one hand, computing an optimal consensus ranking is currently not possible for more than a few dozens of elements. As a consequence, a plethora of heuristics have been proposed whose behaviours are, by essence, difficult to anticipate. On the other hand, 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 optimal consensus.

We are thus facing two challenges. 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. In this paper we tackle these two challenges. Namely we (i) develop 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, and (ii) we provide an approach able to draw frontiers between groups of elements such that their relative order in the whole set of optimal consensus rankings

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

TABLE I
EXAMPLE OF INPUT RANKINGS.

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

are inviolable, thus giving information on the robustness of the positions of elements in the produced consensus ranking.

To illustrate our approach, let us consider the example of Table I above. It has six rankings (r_1 to r_6) of eight elements (A to H). This example is inspired by real use cases we 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, they are tied (*i.e.* ex-aequo), 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.

To compute a consensus ranking, rank aggregation approaches consider pairs of genes and inspect between rankings whether each pair is in the same order (agreement) or not (disagreement).

Let us inspect the set of rankings at a coarse grain level. First, note that D and E are always before all the other genes. A reasonable consensus ranking should then place these two genes at the first two positions. Second A, B and C are always placed before F, G and H. 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 three possible groups of genes. Let us name D, E group 1, A, B, C group 2 and F, G, H group 3 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 within groups should be ranked are the two open questions we want to consider.

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

a) Group 1 (D and E): In four rankings (r_1, r_2, r_5 and r_6), D is tied with E whereas E is before D in r_3 and D is before E in r_4. Placing D before E or E before D in the consensus would be in agreement with only one ranking while placing D and E *ex aequo* in the same bucket is in agreement with four rankings. Placing D and E in the same bucket in the consensus ranking appears to be a very reliable choice.

b) Group 2 (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.

c) Group 3 (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.

This example allows us to highlight two key points. First, it may be the case that solid frontiers appear between several groups of genes. Such frontiers give important information about robustness: D and E should be in the very first positions with no flexibility towards the remaining genes while H can be placed before of after F in a totally arbitrarily way. Second, each group of genes highlighted by the frontiers can be analyzed further, aiming to break the groups into smaller sub-groups (*e.g. group 3*) making easier the computation of a consensus ranking, and maybe enabling the use of an exact algorithm if the obtained sub-groups are small enough.

The purpose of this paper is to introduce a new graph-based pre-process approach for rank aggregation that (i) divides the set of elements into groups where consensus algorithms can be executed independently and (ii) identifies solid frontiers to separate elements. Our approach not only lies in mathematical bases, providing guarantees on the results, but has also been implemented in a real system available to the life science community and tested on various real use cases.

The remainder of this paper is organized as follows. Section II introduces the definitions of the major concepts underlying rank aggregation. Section III proposes graph representations of the input rankings while Section IV introduces an efficient graph-based and reliable-aware consensus ranking procedure able to give information on the robustness of the positions of genes in the consensus ranking produced. Section V first presents ConQuR-BioV2, the tool available to the community where our approach has been fully implemented, and then evaluates our approach on biological queries. Section VI draws conclusions.

II. PRELIMINARIES

In this section we formally introduce the concept of rankings and the problem of rank aggregation.

a) 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 $U = \{A, B, C, D\}$, $r_1 = [\{A\}, \{C\}, \{B\}, \{D\}]$, $r_2 = [\{B, A\}, \{C\}, \{D\}]$ and $r_3 = [\{B, A, C, D\}]$ are 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. 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 have the same position.

b) 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 [6], [7] 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.

c) Distance between two rankings: If r and s are two complete rankings on U (possibly after a unification process), the *Kemeny pseudo-distance*, denoted K, is defined as follows: $K(r, s) = \sum_{\{x,y|x,y \in U\}} \overline{K}_{\{x,y\}}(r, s)$ where $\overline{K}_{\{x,y\}}(r, s) =$

- 1 if x is before y in one ranking whereas y is before x in the other one.
- 1 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: no cost for untying elements which have been tied in the unification bucket.

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 $s^{'}$.

d) 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 $\forall c \in \mathscr{C}$, $S(c^*, R) \leq S(c, R)$.

Note on the complexity of the rank aggregation problem. The rank aggregation problem is known to be NP-hard in most cases so several approximation algorithms and heuristics have been designed [2], [8], [9], [10]. While much research has focused on the case where rankings are permutations (that is, 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 recent works considered the problem of rankings with ties: [8], [10] introduced approximation algorithms to the problem while [11] compared of most of the major heuristics and algorithms adapted to ranking with ties.

e) Optimal consensus and k-frontiers: It is worth noticing that 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 k-frontiers. We define a *k-frontier* as an integer k such that the set of the k first elements is the same in all the optimal consensus.

Illustration. $[\{D, E\}, \{B\}, \{C\}, \{A\}, \{F\}, \{G\}, \{H\}]$ is an optimal consensus of the set of rankings presented in Table I. The associated score is 18 ($2_{\{D,E\}} + 4_{\{A,B\}} + 2_{\{A,C\}} + 2_{\{B,C\}} + 2_{\{F,G\}} + 3_{\{F,H\}} + 3_{\{G,H\}}$). Another possible optimal consensus is $[\{D, E\}, \{B\}, \{C\}, \{A\}, \{H\}, \{G\}, \{F\}]$, with the same score of 18 (by definition). It can be shown that any optimal consensus places D and E before all the remaining elements. In other words, there is a 2-frontier.

f) A pairwise representation: According to the Kemeny pseudo-distance, the cost (regarding a set of rankings R) of placing x before y in the 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 the consensus ranking x before, after or tied with y.

Illustration. Some different pairs (nonexhaustive) of Example I with the associated costs are presented in Table II. For each line, the minimal value is in bold.

For example, if we consider only the pair (D, E), we

TABLE II
PAIRWISE COST MATRIX OF EXAMPLE I.

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
F	H	**3**	**3**	6	3
G	H	**3**	**3**	6	3

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 more "interesting" to tie D and E in a consensus ranking, rather than having D before E or E before D. If we consider only the pair (F,H), we have $before(F,H) = before(H,F) = min(F,H)$ whereas $tied(F,H) > min(F,H)$. We can conclude that it is more "interesting" to have F before H or H before F in a consensus ranking, rather than having F tied with H.

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 cost matrix as follows

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

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.

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 IV describes the process we follow from these structures to compute a reliability-aware consensus.

III. Graph representation for rank aggregation

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

A. Graph-based pairwise representation of the elements

The objective of the graph-based pairwise representations of the elements is to give condensed representations of the pairwise cost matrix presented in Table II to compute a consensus ranking. In these representations, nodes are elements (the genes to rank) and for each pair of elements $\{x,y\}$, the edges are related to which cost(s) in the pairwise cost matrix is (are) minimal among $before(x,y)$, $before(y,x)$ and $tied(x,y)$. As the challenge is both to divide the initial problem into as small sub-problems as possible and to put robust frontiers between groups of elements, we respectively define two graphs: G_e the *graph of elements* and G_r the *robust graph of elements*. In both graphs, an absence of edge from x to y intuitively means that it is reasonable to have y before x in a consensus as $before(y,x)$ is minimal. The difference between the two graphs is that in G_r, the absence of edge from x to y means that $before(y,x)$ is the unique minimum for the pair $\{x,y\}$ whereas the uniqueness is not necessary for G_e.

Let us define formally G_e and G_r.

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

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

Edges in the graph of elements G_e. In G_e, there is **no** edge from x to y if and only if $before(y,x)$ is minimal (not necessarily the unique minimum).

Definition of the robust graph of elements G_r. Let $G_r = (V_r, E_r)$ be the directed graph such that:

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

Edges in the robust graph of elements In G_r, there is **no** edge from x to y if and only if $before(y,x)$ is the *unique* minimum for the pair $\{x,y\}$ (compared to $before(y,x)$ and $tied(x,y)$). In other words, for a pair $\{x,y\}$, we can distinguish three cases:

- case 1: $before(x,y)$ is the unique minimum. Then there is an edge from x to y and no edge from y to x.
- case 2: $before(y,x)$ is the unique minimum. Then there is an edge from y to x and no edge from x to y.
- case 3: if case 1 and case 2 fail, then there are both edges from x to y and from y to x.

Table III 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 edges between x and y in G_e and G_r.

We can see that G_e is a *spanning sub-graph of G_r i.e.* G_e and G_r have the same set of vertices (the elements to be ranked) and each edge of G_e is necessarily also an edge of G_r. The proof is straightforward using Table III. Another information given by Table III is that G_r and G_s provide the same edges for a pair $\{x,y\}$ if and only if there is a unique minimum value between $before(x,y)$, $before(y,x)$ and $tied(x,y)$.

Illustration. Fig. 1 represents both the G_e graph and the G_r graph related to the input rankings presented in Table I.

Comparison of G_e and G_r in Figure 1. Let us focus on the similarities and differences between G_e and G_r. We can see that D, E, A, B, C have exactly the same edges. Interestingly, they are the elements of groups 1 and 2, where no arbitrary choice is possible for any pair of elements (D is tied with E in a strict majority of rankings, A is before B in a strict majority of rankings, ...). Let us now focus on F, G and H (group 3). Element H can be arbitrary placed before or after F (same

TABLE III
Relation between position of min(x,y) and edges in G_e and G_r

$before(x,y)$	$before(y,x)$	$tied(x,y)$	in G_e	in G_r
= min(x,y)	$> min(x,y)$	$> min(x,y)$	$x \rightarrow y$	$x \rightarrow y$
$> min(x,y)$	**= min(x,y)**	$> min(x,y)$	$x \leftarrow y$	$x \leftarrow y$
$> min(x,y)$	$> min(x,y)$	**= min(x,y)**	$x \leftrightarrows y$	$x \leftrightarrows y$
= min(x,y)	$> min(x,y)$	**= min(x,y)**	$x \rightarrow y$	$x \leftrightarrows y$
$> min(x,y)$	**= min(x,y)**	**= min(x,y)**	$x \leftarrow y$	$x \leftrightarrows y$
= min(x,y)	**= min(x,y)**	$> min(x,y)$	$x \quad y$	$x \leftrightarrows y$
= min(x,y)	**= min(x,y)**	**= min(x,y)**	$x \quad y$	$x \leftrightarrows y$

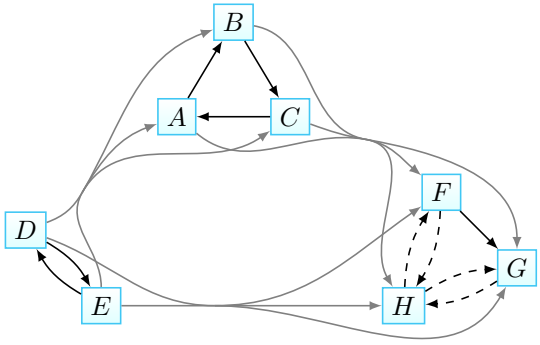

Fig. 1. Graph of elements G_e (solid arcs) and robust graph of elements G_r (solid and dashed arcs) for the example presented in Table I. Grey arcs indicate that all edges from $\{D, E\}$ to $\{A, B, C, F, G, H\}$ and from $\{A, B, C\}$ to $\{F, G, H\}$ are present in both graphs.

with G). This lack of robustness implies that there is an edge from F to H and another from H to F in G_r whereas there is neither an edge from F to H nor from H to F in G_e.

The challenge is now to make use of (i) G_e to divide the initial problem into as many smaller problems as possible and (ii) G_r to reveal the different groups of elements such that there is a natural relative order between groups.

B. Graph of 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 and G_r.

a) Definitions: 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).

Let now $G = (V, E)$ be a graph and $\mathscr{C} = \{c_1, ..., c_z\}$ be the set of the strongly connected components of G. The **graph of strongly connected components** of G is the directed graph $G' = (V', E')$ such that (1) V' = $\mathscr{C}$ and (2) E' = $\{(c_i, c_j) \in \mathscr{C}^2 : i \neq j \wedge \exists x \in c_i, y \in c_j : (x, y) \in E\}$.

The vertices of G' are the strongly connected components of G. There is an edge from a vertex c_i to another vertex c_j in G' 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 edge of G. Computing the strongly connected components of G can be done with Tarjan's strongly connected components algorithm [12].

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

In this article, we respectively denote G_e^c and G_r^c the graph of the strongly connected components of G_e and G_r.

b) Interest of G_r^c: Vertices of G_r^c are groups of elements such that there is a natural relative order between them: there is a unique topological sort of G_r^c as for two vertices v_1, v_2 of G_r^c, there is necessarily an edge from v_1 to v_2 or an edge from v_2 to v_1 (in G_r, two elements are necessarily connected by at least one edge, see Table III). Conversely, if we consider elements belonging to the same vertex of G_r^c, the three following situations are possible (nonexclusive). First, some pairs $\{x, y\}$ have $tied(x, y)$ as unique minimal cost and should not be separated. Second, there are incompatibilities (e.g., $before(x, y)$, $before(y, z)$ and $before(z, x)$ are respectively the unique minimum for $\{x, y\}$, $\{y, z\}$ and $\{z, x\}$). Third, there is a lack of robustness (two relative placements have the same minimal cost for a pair $\{x, y\}$).

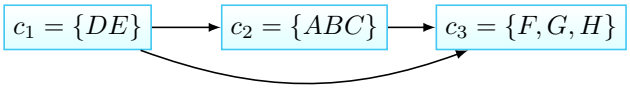

Fig. 2. G_r^c, the graph of strongly connected components of G_r.

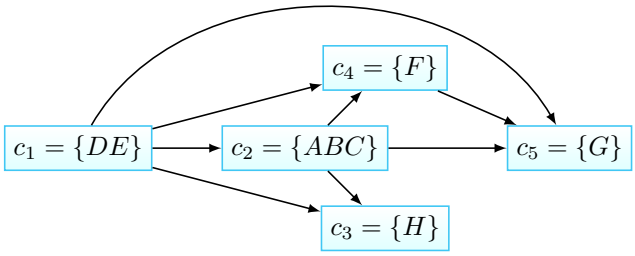

Fig. 3. G_e^c, the graph of strongly connected components of G_e.

Illustration. Fig. 2 represents the G_r^c graph of Example I. G_r^c has three edges (G_r has three SCCs) which correspond to the groups 1, 2, 3. Computing the SCCs of G_r enables the gathering of (i) $\{D, E\}$ which should not be separated as they are tied in a strict majority of rankings, (ii) $\{A, B, C\}$ which form an incompatibility difficult to solve, and (iii) $\{F, G, H\}$ which may not be split any further in this graph.

c) Interest of G_e^c: To compute efficiently a consensus, we need groups to be as small as possible, and G_e^c can be very useful to break some groups of elements built by G_r^c. Indeed, as G_e is a spanning sub-graph of G_r, some edges of G_r may have been removed. The consequence is that smaller groups may appear, still under the condition that a missing edge is synonym of a minimal cost for a "before relation".

Illustration. Fig. 3 represents graph G_e^c of Example I. We can see that vertex c_3 of G_r^c has been broken into three sub-groups. This break was enabled by the possibility to place H before or after F and G.

IV. Reliability-aware rank aggregation

This section introduces fundamental properties on the graph representation presented in the above section which form the basis of a new rank aggregation procedure to efficiently provide reliability-aware consensus rankings.

A. Fundamental properties

We present two properties in this section. The first one is related to G_e^c and can be exploited to divide the initial problem

into several sub-problems, which is important to compute a high-quality consensus ranking in a reasonable time. The second one is related to G_r^c and can be exploited to set robust frontiers between groups of elements.

Let us set two points of vocabulary. First, a consensus ranking c *respects* a topological sort $\mathscr{T} = [T_1, T_2, ..., T_k]$ of G_r^c or G_e^c if and only if for all pairs of elements $\{x, y\}$ such that $x \in T_i$, $y \in T_j$ and $i < j$, x is before y in c. Last, 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 I and consider c_1 the SCC of G_s 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 $r'_1 := [\{D, E\}]$, $r'_2 := [\{D, E\}]$, $r'_3 := [\{E\}, \{D\}]$, $r'_4 := [\{D\}, \{E\}]$, $r'_5 := [\{D, E\}]$, $r'_6 := [\{D, E\}]$.

Concatenation property. Let $\mathscr{T} = [T_1, T_2, \ldots, T_k]$ be a topological sort of G_e^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 (1) 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 edge 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 can not be higher than the cost induced by (x, y) in c. Finally, $S(\mu, R) \leq S(c, R)$ as claimed.

Corollary. There exists an optimal consensus ranking compatible with any topological sort $\mathscr{T} = [T_1, ..., T_k]$ of G_e^c.

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 an algorithm we present below.

Robust frontiers property. Any optimal consensus respects the unique topological sort $\mathscr{T} = [T_1, ..., T_k]$ of G_r^c.

Proof. Consider a consensus ranking c which does not respect $\mathscr{T}$. We will show that c is not optimal. From c, we build a consensus c' as follows: take first the elements of T_1 in the same order as they are in c, then append the elements of T_2 in the same order as they are in c, and repeat the operation for the elements of T_3, ..., until T_k. By construction, c' respects $\mathscr{T}$, thus is different from c. Now, let us compare $S(c, R)$ and $S(c', R)$ using (1). Each pair of elements (x, y) such that x and y are in the same relative order in c and c' induce 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 $\mathscr{T}$ *i.e* in two different SCCs of G_r. Let set $x \in T_i$ and $y \in T_j$ with $i < j$. As T_i is before T_j, there is no edge from y to x in G_r. By construction of G_r, we can conclude that $before(x, y) = min(x, y)$, $before(y, x) > min(x, y)$ and $tied(x, y) > min(x, y)$. As a consequence, $S(c, r) > S(c', R)$ *i.e.* c can not be an optimal consensus.

Illustration. Consider R the set of rankings presented in Table I and consider c_1 the SCC of G_r^c which contains D and E. We obtain that in any optimal consensus ranking, D and E are before all the remaining elements and A, B and C are before F, G and H. There is a 2-frontier and a 5-frontier.

B. Reliability-aware rank aggregation

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

The question is now to determine 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 auxilliary ranking algorithm (depending on the size of SCC, an exact algorithm or rather an heuristic) should be called to provide a consensus ranking.

Illustration. Let us inspect again the example presented in Table I. A possible topological sort for G_r is: $[c_1, c_2, c_4, c_5, c_3]$ *i.e.* $[\{D, E\}, \{A, B, C\}, \{F\}, \{G\}, \{H\}]$. 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) $[\{F\}]$, (iii) $[\{G\}]$ and (iv) $[\{H\}]$ are optimal consensus rankings for the sub-problem induced respectively by (i) c_1, (ii) c_4, (iii) c_5 and (iv) c_3 (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) A, B and C share the 3^{rd}, 4^{th} and 5^{th} position (still not ordered yet); (iii) F is in 6^{th} position; (iv) G is in 7^{th} position; (v) H is in 8^{th} position.

Suppose now that the algorithm used to solve the sub-problem induced by c_2 placed A before both B and C and placed B before C. Then, the obtained consensus ranking for the example presented in section I is $[\{D, E\}, \{A\}, \{B\}, \{C\}, \{G\}, \{F\}, \{H\}]$.

Continuing with our example, let us 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_2 of Fig 3, 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.

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

Input: R : set of rankings
begin
  $M \leftarrow$ PairwiseCostMatrix(R)
  $G_r \leftarrow$ ComputeGraphGr(M)
  $G_e \leftarrow$ ComputeGraphGe(M)
  $G_{cr} \leftarrow$ SCC(G_r)
  $G_{ce} \leftarrow$ SCC(G_e)
  topolSortGr $\leftarrow$ TopologicalSorting(G_{cr})
  topolSortGe $\leftarrow$ TopologicalSorting(G_{ce})
  nbSccGe $\leftarrow$ numberSCCofGe
  nbSccGr $\leftarrow$ numberSCCofGr
  consensus $\leftarrow$ EmptyListOfSets()
  placed $\leftarrow 0$
  $j \leftarrow 1$
  nextFrontier $\leftarrow$ topolSortGr[j].length()
  for $i \leftarrow 1$ **to** *nbSccGe* **do**
    if $\exists v1 \neq v2$ **in** *topolSortGe[i]* **with** $tied(v1, v2) > min(v1, v2)$ **then**
      inducedInput $\leftarrow$ CopyInput(R)
      Remove from inducedInput all the elements not in topolSortGe[i]
      subConsensus $\leftarrow$ auxiliaryAlgo(inducedInput)
      Append each bucket of subConsensus to consensus
    else
      Append topolSortGe[i] to consensus
    placed $\leftarrow$ placed + topolSortGe[i].length()
    if *placed* $=$ *nextFrontier* ***and*** $j < nbSccGr$ **then**
      Add a k-frontier in the consensus
      $j \leftarrow j + 1$
      nextFrontier $\leftarrow$ nextFrontier + topolSortGr[j]
Result: consensus with k-frontiers

Algorithm 1 describes the procedure providing a consensus ranking with k-frontiers. Note that the result of the returned consensus ranking may depend on the auxiliary algorithm used to solve the sub-problems. Moreover, if we never call an heuristic (either if case 1 always holds or if an exact algorithm is used instead), then the consensus provided is optimal.

V. Evaluation

This section introduces ConQuR-BioV2, a tool in which our approach has been fully implemented and which is concretely in-use for the life science community. Then, we present the experimental setting we set-up followed by the results we obtained on real large biological datasets.

A. ConQuR-BioV2: a new rank aggregation tool for bio data

ConQuR-Bio has been initially introduced in [5] to guide users in the maze of biological data available in public biological databases. ConQuR-Bio processes in three steps. First, given a keyword, ConQuR-Bio queries a set of databases able to provide synonyms of the user keyword listed in medical thesaurus from the UMLS (e.g., MeSH, OMIM, SNOMED CT, ICD9CM). Second, for each synonym (and the initial keyword), ConQuR-Bio automatically generates a query – also called a a *reformulation* – and send it to the NCBI Gene database. For each reformulation, ConQuR-Bio collects a set of answers, ranked by relevance (number of occurrences of the keyword in the answer). Third, ConQuR-Bio uses rank aggregation techniques to provide a final consensus ranking.

In the example provided in Figure 4 the user has considered the keyword *Long QT syndrome* for which 48 synonyms have been found (e.g., Timothy syndrome). The set of answers can be visualized by the user.

ConQuR-Bio is used by several members of the life science community demonstrating the need for this kind of tools. However, it suffered from two main weaknesses. First, the number of elements obtained as answer to a query is constantly augmenting, there is thus a crucial need for an approach able to scale. Second, ConQuR-Bio is 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 graph-based pre-process introduced in Section IV and (ii) the user interface has been adapted to exploit the information on the k-frontiers produced.

B. Evaluation: Experimental setting

a) Datasets: Based on the expertise of our life science collaborators (from Institut Curie, Institut Pasteur, Orphanet and the Childrens' Hospital of Philadelphia), we selected a set of 30 diseases for which there exist relationships between diseases and genes. This list provided in Table IV includes cancers (e.g., breast cancer), rare diseases (e.g., Beckwith-Wiedemann syndrome) and childhood diseases (e.g., Long QT syndrome).

For each disease, we used ConQuR-Bio to find different synonyms and thus generate reformulations. Each reformulation of a query provides a ranking (of answers). As shown in Table IV, each disease (denoted by a initial keyword which is always a MeSH term) is associated with 14 synonyms in average (from 3 to 48 synonyms). For each disease, 322 genes by rankings are provided in average (in all the reformulations).

b) Algorithms: Several studies of rank aggregation algorithms have been performed in the past ten years. One of the most recent is [11] 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 [9], Copeland method [13] and BioConsert [7]. We thus considered the exact algorithm and the three heuristics above that we all slightly adapted to handle the pseudo-distance described in Section II (to manage rankings that have been completed by unification).

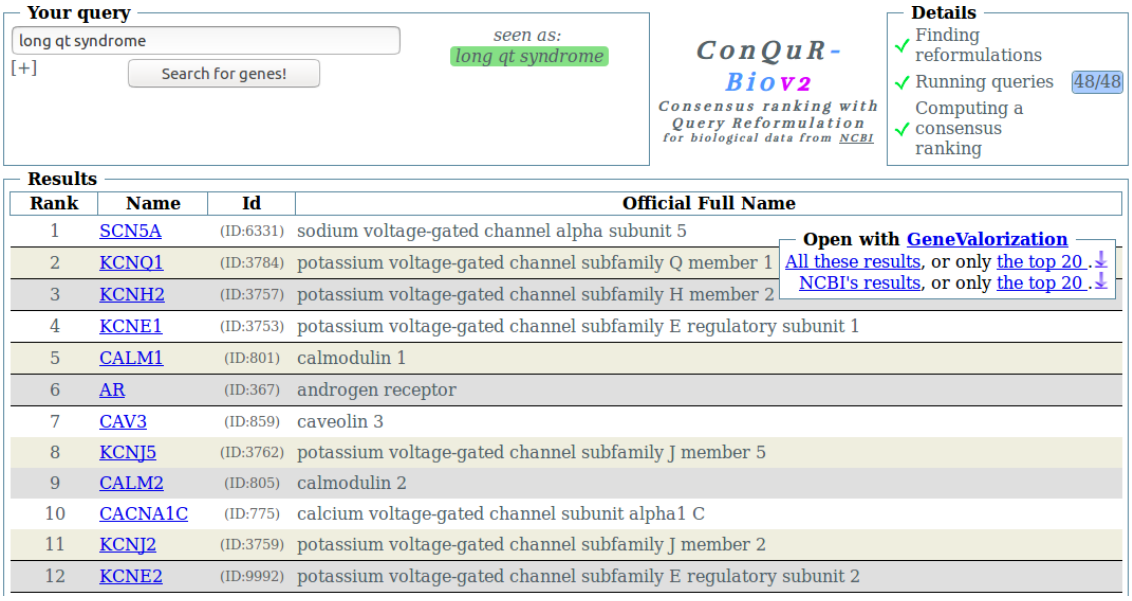

Fig. 4. Interface of ConQuR-BioV2

TABLE IV
DATASETS TABLE.

ID	disease	# rankings	# genes
1	ADHD	28	350
2	Alagille syndrome	24	382
3	Angelman syndrom	9	239
4	Beckwith-Wiedemann syndrome	19	272
5	Bladder cancer	15	556
6	Breast cancer	14	648
7	Cervix cancer	13	437
8	Colorectal cancer	5	379
9	Crouzon syndrome	23	423
10	Darier disease	20	167
11	DiGeorge syndrom	29	258
12	Ehlers-Danlos syndrome	15	121
13	Familial ataxia	22	212
14	Fanconi anemia	18	355
15	Leber congenital amaurosis	17	309
16	Long QT syndrome	48	236
17	Morquio syndrome	35	423
18	Mucoviscidiosis	10	352
19	Myotonic dystrophy	16	207
20	Neuroblastomas	3	279
21	Noonan syndrome	23	387
22	Omenn syndrome	9	367
23	Paget disease of bone	12	270
24	Polycystic kidney disease	9	262
25	Retinoblastoma	17	437
26	Sandhoff disease	22	206
27	Sideroblastic anemia	11	249
28	Tangier disease	22	295
29	Usher syndrome	24	258
30	Wilms tumor	21	315

c) 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* [14], [6] which consists in normalizing the distance to show the additional disagreement one solution has (c) compared to an optimal solution (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. 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* [11]: $m\text{-}gap(c,R) = \frac{S(c,R)}{S(c^+,R)} - 1$. Note that if an optimal solution is found $m\text{-}gap(c,R) = gap(c,R)$.

C. Evaluation on biological datasets

We now consider five experiments performed on the 30 biological datasets described above. The first three are related to the evaluation of the pre-process approach while the last two focus on the robustness of the consensus ranking obtained (k-frontiers).

a) Experiment 1 (Interest of the pre-process): The first experiment aims at evaluating the role of the pre-process in the final ranking provided. Figure 5 shows that a large amount of genes can be ranked by the pre-process without any need of any algorithms or heuristics. More precisely (i) for 10 datasets, an optimal solution was found (100% of the genes have been placed by the pre-process), (ii) in the remaining datasets, 43% to 97% of genes are positioned this way. This represents for each query a range of 101 to 423 elements (in average 249) positioned by the pre-process on its own.

To rank the remaining set of the elements, there are two possibilities. The first (and obviously preferred when possible)

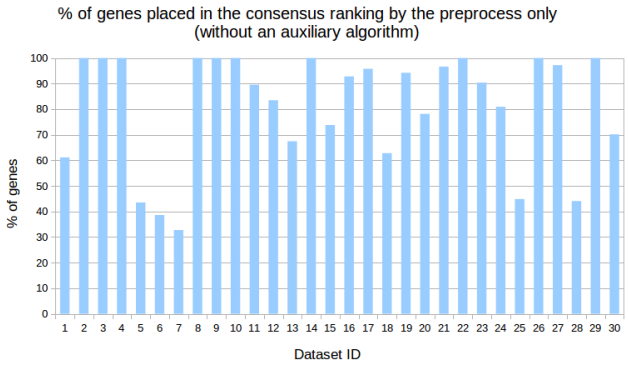

Fig. 5. Proportion of genes placed in the consensus ranking by the pre-process only for each biological dataset.

is to use an exact algorithm which provides best quality results but may be impossible if the set of elements to rank is too large. The second one (default) is to use a heuristic. The next experiment investigates the ability to use the exact algorithm.

b) Experiment 2 (use of the exact algorithm): The current implementation of Algorithm 1 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 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 (several seconds). This second experiment provides the number of genes that are placed (i) by the pre-process (cf experiment 1), (ii) then by an exact algorithm and (iii) then by a heuristic. The result is shown in Figure 6. Thanks to the exact algorithm, optimal solutions have been found for 14 more datasets (a total of 24 when adding the optimal solutions found by the pre-process). A heuristic has to be used for only 6 remaining datasets. Determing the best heuristic to choose is a key point that we consider in the next experiment.

c) Experiment 3 (Benching various heuristics with/out pre-processing): In this third experiment we compare the m-gap and the computation time of the three heuristics mentioned in subsection V-B, used with and without pre-processing.

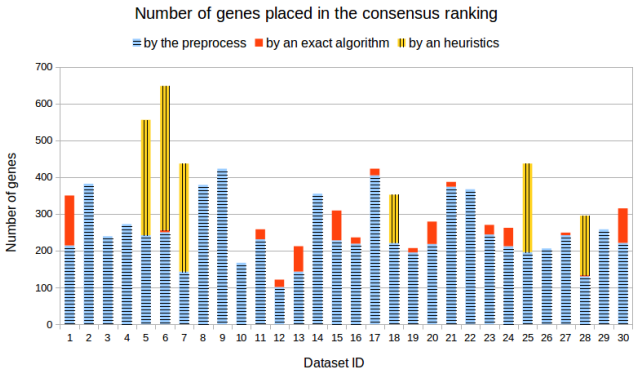

Fig. 6. Number of genes placed in the consensus ranking by (i) the pre-process, (ii) an exact algorithm, (iii) a heuristic.

TABLE V
BENCHMARK

heuristics	mean m-gap	mean time	max time
$CopelandMethod$	$1.49 * 10^{-1}$	11.54 ms	40.23 ms
$pre\text{-}process$ $+CopelandMethod$	$1.02 * 10^{-2}$	29.22 ms	142.73 ms
$KwikSort$	$1.03 * 10^{-2}$	0.36 ms	0.93 ms
$pre\text{-}process$ $+KwikSort$	$3.86 * 10^{-3}$	27.46 ms	78.44 ms
$BioConsert$	$2.81 * 10^{-5}$	141.37 ms	584.02 ms
$pre\text{-}process$ $+BioConsert$	$6.32 * 10^{-5}$	71.3 ms	306.32 ms
$pre\text{-}process$ $+exact(SCC < 80)$ $+BioConsert$	$3.64 * 10^{-6}$	2.53 s	18s (1 dataset) $\leq$ 7s otherwise

Table V provides the results of the bench we conducted. Two points can be noticed. First, using the pre-process 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 63.0% and the m-gap of CopelandMethod has been decreased by 93%. Computation time remains very reasonable (less than 150 milliseconds).

Second, using pre-processing allows to increase the speed of high-quality algorithms still improving their quality. More precisely, the computation time of BioConsert has been reduced by 50% while its m-gap has been reduced by 87%.

Based on these experiments, the implementation of ConQuR-BioV2 runs systematically the exact algorithm and BioConsert in parallel. If the exact algorithm gives the result in less than 2 seconds, this result is used. Otherwise, the result of BioConsert is used. More generally, our pre-process allows us to run in parallel auxiliary algorithms on different connected components.

d) Experiment 4: Robust graph of elements on biological datasets: In this last but not least series of experiments, we provide information on the k-frontiers between elements in real datasets. Figure 7 indicates the number of datasets for which the elements in the first k positions are robust, that is, these elements are positioned in the first k positions by any optimal consensus. Out of the 30 datasets, 25 have a 1-frontier, that is, one single gene is (unambiguously) known to be the most associated with the disease (all the possible optimal consensus placed it first). Twelve datasets have a 4-frontier, that is, four genes are strongly known to be the more associated with the disease. Unsurprisingly, only a few datasets have a 10-frontier.

Finally, five datasets do not have any element which position is always the same in the set of all the possible solutions. In these diseases, several genes may be currently candidates to be associated with the disease but still under consideration or not associated with all the forms of the disease.

As a conclusion of this experiment, our approach here helps users distinguishing a possible ranking (possibly provided

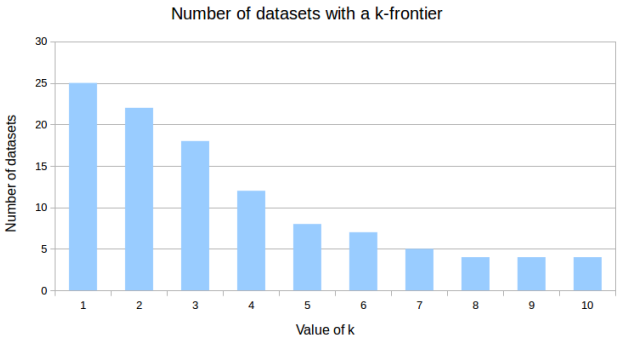

Fig. 7. Number of datasets with a k-frontier

somehow arbitrarily) from a robust highly-reliable ranking.

e) Experiment 5 (qualitative): Using k-frontiers: On Figure 4, we can see that for the Long QT syndrome, the user has access to precise information on the robustness with eight frontiers indicated. More precisely, such frontiers allow the user to know that the order between the six first elements is robust (e.g., CALM1 is strictly less important than KCNE1 wrt Long QT syndrome). As for the elements ranked 7th to 11th, there may exist other optimal consensus where their relative positions differ. Thus, if an experiment should be conducted on CAV3 (positioned 7th) then the experiment should consider also genes at positions 8 to 11 as their relative order is not robust. The interface thus provides a guide to the user on further research to be conducted.

VI. Conclusion

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. Our solution lies on rank aggregation techniques. Our first contribution consists in introducing 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, ensuring that 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 on a reasonable time (less than a second) for several hundreds of elements. Our second contribution defines the concept of k-frontier to guide users in their understanding of the consensus by highlighting robust positions of elements.

The originality of our approach is twofold. First, it is based on graph representations allowing to better capture the intrinsic relationships between elements of the rankings. It can be combined with all the heuristics currently available and discussed in [11]. Second, our approach is reliability-aware, that is, it provides high-quality results possibly using an exact algorithm while providing users with a clear information on the robust elements in the consensus ranking produced.

As seen from the introduction, a plethora of heuristics have been proposed to the rank aggregation problem [2], [7], [9], [10], [13], compared in various studies [6], [11], [14]. New divide-and-conquer kind of heuristics have been introduced even recently (see for example [15], [16]). However, our approach differs from such approaches in that we provide strong guarantees on the result: our decomposition in subproblems necessarily complies with an optimal solution. As for the reliability-aware aspect of our contribution, we are not aware on any work considering the problem of several optimal solutions in a context with real datasets and users and studying the robustness of the results.

In the future, we will work with the ConQuR-BioV2 users to collect their feedback. Ongoing work includes extending our approach to other kinds of 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.

References

[1] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova, "Entrez gene: gene-centered information at ncbi," *Nucleic acids research*, vol. 33, no. suppl_1, pp. D54–D58, 2005.

[2] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proc. of the WWW conference*. ACM, 2001, pp. 613–622.

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

[4] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "An algorithmic view of voting," *SIAM J. Discrete Math.*, vol. 30, pp. 1978–1996, 2016.

[5] B. Brancotte, B. Rance, A. Denise, and 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] F. Schalekamp and A. van Zuylen, "Rank aggregation: Together we're strong," in *Proc of ALENEX*, 2009, pp. 38–51.

[7] S. Cohen-Boulakia, A. Denise, and S. Hamel, "Using medians to generate consensus rankings for biological data," in *SSDBM 2011: Scientific and Statistical Database Management Conference*, 2011.

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

[9] N. Ailon, M. Charikar, and A. Newman, "Aggregating inconsistent information: Ranking and clustering," *J. ACM*, vol. 55, no. 5, pp. 23:1–23:27, Nov. 2008.

[10] N. Ailon, "Aggregation of partial rankings, p-ratings and top-m lists," *Algorithmica*, vol. 57, no. 2, pp. 284–300, Feb. 2010.

[11] B. Brancotte, B. Yang, G. Blin, S. Cohen Boulakia, A. Denise, and S. Hamel, "Rank aggregation with ties: Experiments and analysis," *Proc. of the VLDB Endowment (PVLDB)*, vol. 8, no. 11, p. 2051, Aug 2015.

[12] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, 1972.

[13] A. H. Copeland, "A reasonable social welfare function," in *Univ. of Michigan Seminar on Appl. of Mathematics to the social sciences*, 1951.

[14] A. Ali and M. Meil, "Experiments with kemeny ranking: What works when?" *Mathematical Social Sciences*, vol. 64, no. 1, pp. 28 – 40, 2012, computational Foundations of Social Choice.

[15] J. Ding, D. Han, J. Dezert, and Y. Yang, "A new hierarchical ranking aggregation method," *Information Sciences*, vol. 453, pp. 168 – 185, 2018.

[16] P. S. Badal and A. Das, "Efficient algorithms using subiterative convergence for kemeny ranking problem," *Computers & Operations Research*, vol. 98, pp. 198 – 210, 2018.