Passage à l'échelle, propriétés et qualité des algorithmes de classements consensuels pour les données biologiques massives

Pierre Andrieu

▶ **To cite this version:**

Pierre Andrieu. Passage à l'échelle, propriétés et qualité des algorithmes de classements consensuels pour les données biologiques massives. Bio-informatique [q-bio.QM]. Université Paris-Saclay, 2021. Français. NNT : 2021UPASG041 . tel-03335281

HAL Id: tel-03335281
https://theses.hal.science/tel-03335281

Submitted on 6 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

Passage à l'échelle, propriétés et qualité des algorithmes de classements consensuels pour les données biologiques massives

Scalability, properties and quality of consensus ranking algorithms for massive biological data

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Unité de recherche : Université Paris-Saclay, CNRS, Laboratoire interdisciplinaire des sciences du numérique, 91405, Orsay, France
Référent : Faculté des sciences d'Orsay

Thèse présentée et soutenue à Paris-Saclay, le 14/06/2021, par

Pierre ANDRIEU

Composition du Jury

Gaëlle LELANDAIS Professeure, I2BC (Institut de biologie intégrative de la cellule), Université Paris-Saclay	Présidente
Guillaume FERTIN Professeur, LS2N (Laboratoire des Sciences du Numérique de Nantes), Université de Nantes	Rapporteur & Examinateur
Sylvie HAMEL Professeure, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada	Rapporteur & Examinatrice
Mokrane BOUZEGHOUB Professeur, DAVID (Laboratoire Données et Algorithmes pour une Ville Intelligente et Durable), UVSQ, Université Paris-Saclay	Examinateur
Miguel COUCEIRO Professeur, LORIA (Laboratoire lorrain de Recherche en Informatique et ses Applications), Université de Lorraine	Examinateur
Stéphane VIALETTE Directeur de Recherche CNRS, LIGM (Laboratoire d'Informatique Gaspard-Monge), Université Gustave Eiffel	Examinateur

Direction de la thèse

Sarah COHEN-BOULAKIA Professeure, LISN (Laboratoire Interdisciplinaire des Sciences du Numérique), Université Paris-Saclay	Directrice de thèse
Alain DENISE Professeur, LISN (Laboratoire Interdisciplinaire des Sciences du Numérique), Université Paris-Saclay	Co-Encadrant
Adeline PIERROT Maître de conférences, LISN (Laboratoire Interdisciplinaire des Sciences du Numérique), Université Paris-Saclay	Invitée

Thèse de doctorat

NNT : 2021UPASG041

Table des matières

Table des figures

Liste des tableaux

Introduction

Nous sommes depuis plus de deux décennies dans l'ère du *big data* dont l'émergence a été facilitée par d'importantes évolutions technologiques. La capacité de stockage en particulier a été considérablement augmentée, passant du mégaoctet dans les années 1990 au gigaoctet dans les années 2000 pour atteindre aujourd'hui le téraoctet [28, 72]. Une conséquence directe est la quantité de plus en plus démesurée de données qu'il est possible de rendre accessible publiquement. Parallèlement, en biologie, l'avènement du séquençage haut débit a entraîné une baisse significative du coût de séquençage et ainsi une nette augmentation du nombre de séquences déterminées (d'ADN ou d'ARN). À titre d'exemple, alors qu'il fallut dix ans et plusieurs milliards de dollars pour le premier séquençage du génome humain, il suffit désormais d'une semaine et de quelques milliers de dollars pour aboutir au même résultat. L'essor d'approches pluridisciplinaires a considérablement accéléré le traitement et l'analyse de ces données en impliquant notamment les mathématiques et l'informatique (modélisation, algorithmique, ...). Ainsi, les 20 dernières années ont vu une augmentation très importante des données associées aux séquences biologiques. Sont notamment concernées la biologie moléculaire (compréhension des mécanismes de réplication de l'ADN, de transcription d'ADN en ARNm, de traduction d'ARNm en protéines), la biologie structurale (structures d'ARN, structures protéiques, ...), la génétique (mutations, phylogénie, ...), la médecine (associations gènes-maladies, cibles thérapeutiques, ...).

Accès aux données biologiques. Regrouper et organiser ces données sont donc devenus des enjeux très importants pour la mise à disposition et l'approfondissement des connaissances. De nombreuses bases de données biologiques sont créées pour tenter de répondre à ce besoin. Le journal *Nucleic Acids Research* (NAR), qui en répertorie chaque année une part importante, comptabilise plus d'un millier de références dont 89 nouvelles pour la seule année 2021 [77]. Mais ces nouvelles données biologiques ne sont pas seulement très nombreuses, elles sont aussi de natures diverses à plusieurs niveaux [31]. Elles sont à la fois hétérogènes (séquences nucléiques et protéiques, interactions entre entités biologiques, lien entre gènes et maladies, conformation 3D d'ARN et protéines, variants génétiques...) et dépendantes de contextes biologiques (pathologie, espèce, étape du développement...). Pour répondre à cette problématique, les bases de données biologiques s'organisent en deux types : les bases généralistes constituées selon la nature des données à stocker et les bases de données spécialisées autour de

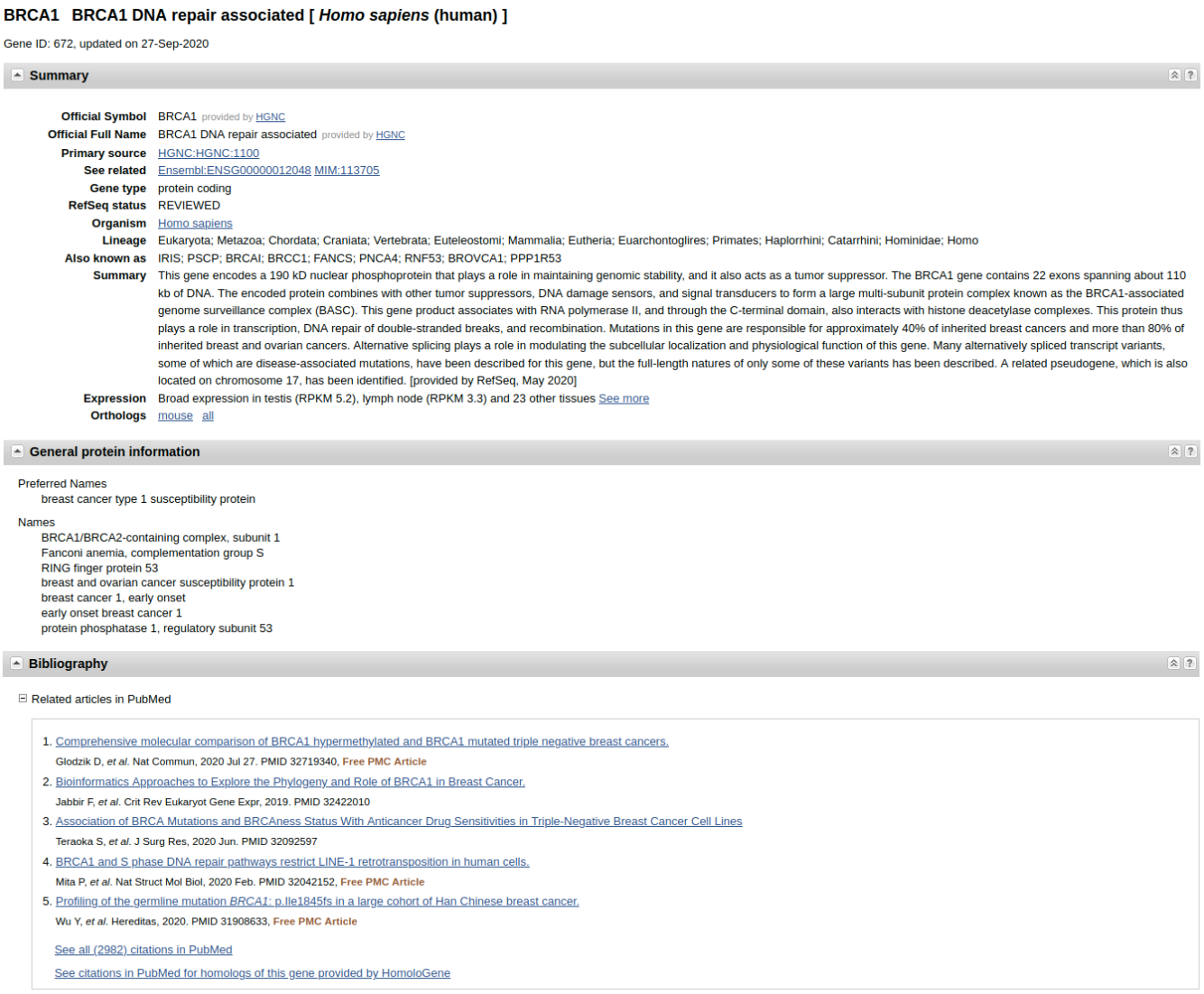
thématiques biologiques. La modélisation des objets biologiques varie parfois beaucoup d'une base à l'autre, et les données sont associées à des degrés de qualité variables. Les bases de données "thématiques" garantissent souvent une plus grande fiabilité des données par rapport aux bases généralistes, grâce notamment à une vérification manuelle du contenu. La contrepartie est que la quantité d'information peut être insuffisante et qu'il faut parfois la compléter en utilisant d'autres bases.

La diversité et la complémentarité des bases de données biologiques a de ce fait rendu indispensable l'utilisation de portails permettant un accès centralisé à ces nombreuses données hétérogènes, au point que des organisations gouvernementales sont dédiées à cette tâche depuis une vingtaine d'années : le *National Center for Biotechnology Information* (NCBI) aux États-Unis d'Amérique, le *European Bioinformatics Institute* (EBI) en Europe et la *DNA Data Bank of Japan* (DDBJ) au Japon.

Ces portails tirent profit de la complémentarité des bases de données et de l'abondance des données en exploitant des références établies entre elles à deux niveaux. Premièrement, des références croisées sont établies entre entités de bases de données différentes. Ainsi, tel qu'illustré par l'extrait d'une fiche d'une base de données génétique en Figure 1, la fiche d'un gène codant une protéine contient un lien vers l'entrée correspondant à cette protéine au sein d'une base de données protéique. Deuxièmement, les éléments d'une même base de données peuvent être connectés entre eux grâce à des algorithmes. Par exemple, en consultant une séquence nucléotidique, on a la possibilité de retrouver celles qui lui "ressemblent". De la même manière, à partir d'un article, on peut trouver d'autres articles provenant de la même base de données portant sur le "même" sujet grâce à une étude sémantique des entrées (mots en commun, fréquence et proximité de ces mots, ...). Lorsqu'un utilisateur interroge une base de données avec un mot clé, une analyse de texte est également à l'oeuvre : le mot clé est recherché dans les différentes entrées de la base permettant un tri par pertinence. Par exemple, interroger la base de données *Gene* du NCBI [66] avec pour mot clé un nom de maladie permet d'établir des associations gènes-maladies.

Utilisation de mots-clés synonymes pour enrichir l'information apportée par une base de données. Avec la recherche de mots clés au sein des fiches apparaît une problématique récurrente en biologie : comment faire lorsque des travaux de recherche utilisent des dénominations différentes pour désigner un même gène ou une même maladie ? Une conséquence de ce problème est qu'une analyse sémantique au sein d'une base de données de gènes peut aboutir à des résultats très différents en fonction de la dénomination choisie pour lancer la recherche.

Prenons l'exemple de médecins qui s'interrogent sur les gènes les plus impliqués dans le cancer du sein. Pour cela, ils vont interroger la base de données *Gene* du *NCBI* avec le mot clé *Breast cancer* (les premiers résultats de cette requête sont visibles en Figure 2). La base de données renvoie une liste de 19567 gènes dont 4433 pour l'espèce humaine classés par ordre de pertinence. Nous pouvons voir sur la Figure 2 que le gène *BRCA2* apparaît en première position, le gène *BRCA1* en deuxième position et ainsi de suite. La pertinence d'un gène donné vis-à-vis du mot clé (*Breast Cancer* pour notre exemple) est déterminé par le *NCBI* par le nombre de fois que le mot



BRCA1 **BRCA1 DNA repair associated [*Homo sapiens* (human)]**

Gene ID: 672, updated on 27-Sep-2020

Summary

Official Symbol	BRCA1 provided by HGNC
Official Full Name	BRCA1 DNA repair associated provided by HGNC
Primary source	HGNC:HGNC:1100
See related	Ensembl:ENSG00000012048 MIM:113705
Gene type	protein coding
RefSeq status	REVIEWED
Organism	Homo sapiens
Lineage	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo
Also known as	IRIS; PSCP; BRCAI; BRCC1; FANCS; PNCA4; RNF53; BROVCA1; PPP1R53
Summary	This gene encodes a 190 kD nuclear phosphoprotein that plays a role in maintaining genomic stability, and it also acts as a tumor suppressor. The BRCA1 gene contains 22 exons spanning about 110 kb of DNA. The encoded protein combines with other tumor suppressors, DNA damage sensors, and signal transducers to form a large multi-subunit protein complex known as the BRCA1-associated genome surveillance complex (BASC). This gene product associates with RNA polymerase II, and through the C-terminal domain, also interacts with histone deacetylase complexes. This protein thus plays a role in transcription, DNA repair of double-stranded breaks, and recombination. Mutations in this gene are responsible for approximately 40% of inherited breast cancers and more than 80% of inherited breast and ovarian cancers. Alternative splicing plays a role in modulating the subcellular localization and physiological function of this gene. Many alternatively spliced transcript variants, some of which are disease-associated mutations, have been described for this gene, but the full-length natures of only some of these variants has been described. A related pseudogene, which is also located on chromosome 17, has been identified. [provided by RefSeq, May 2020]
Expression	Broad expression in testis (RPKM 5.2), lymph node (RPKM 3.3) and 23 other tissues See more
Orthologs	mouse all

General protein information

Preferred Names
breast cancer type 1 susceptibility protein

Names
BRCA1/BRCA2-containing complex, subunit 1
Fanconi anemia, complementation group S
RING finger protein 53
breast and ovarian cancer susceptibility protein 1
breast cancer 1, early onset
early onset breast cancer 1
protein phosphatase 1, regulatory subunit 53

Bibliography

Related articles in PubMed

1. Comprehensive molecular comparison of BRCA1 hypermethylated and BRCA1 mutated triple negative breast cancers.
   Glodzik D, *et al.* Nat Commun, 2020 Jul 27. PMID 32719340, Free PMC Article
2. Bioinformatics Approaches to Explore the Phylogeny and Role of BRCA1 in Breast Cancer.
   Jabbir F, *et al.* Crit Rev Eukaryot Gene Expr, 2019. PMID 32422010
3. Association of BRCA Mutations and BRCAness Status With Anticancer Drug Sensitivities in Triple-Negative Breast Cancer Cell Lines
   Teraoka S, *et al.* J Surg Res, 2020 Jun. PMID 32092597
4. BRCA1 and S phase DNA repair pathways restrict LINE-1 retrotransposition in human cells.
   Mita P, *et al.* Nat Struct Mol Biol, 2020 Feb. PMID 32042152, Free PMC Article
5. Profiling of the germline mutation *BRCA1*: p.Ile1845fs in a large cohort of Han Chinese breast cancer.
   Wu Y, *et al.* Hereditas, 2020. PMID 31908633, Free PMC Article

See all (2982) citations in PubMed

See citations in PubMed for homologs of this gene provided by HomoloGene

FIGURE 1 – Extrait de la fiche d'annotation du gène *BRCA1* au sein de la base de données *Gene* du NCBI.

clé est présent dans la fiche du gène. Il existe un cas de figure particulier : des gènes peuvent partager une même position dans un classement, ils sont alors considéré ex-aequo en terme de pertinence.

Malheureusement ce système est biaisé en raison du problème de nomenclature soulevé plus haut. Ainsi, en pratique, il est fréquent que certains gènes très pertinents et donc théoriquement attendus en tête de liste se retrouvent mal classés voire absents du classement retourné. Afin de pallier ce problème, les médecins interrogent la base de données à plusieurs reprises en utilisant différentes dénominations synonymes de la maladie qu'on appelle *reformulations*. Ainsi, le gène *MUC1* qui apparaît en positon 170 en utilisant le mot clé *Breast cancer* se trouve dans le top-10 lorsqu'on utilise la reformulation *Breast carcinoma* ou encore *Malignant neoplasm of breast*. Il est important de noter que les différents classements de gènes obtenus ne contiennent pas le même nombre de gènes (6129 pour *Breast carcinoma* dont 3321 pour l'espèce humaine par exemple) et que l'ensemble des gènes contenus dans les "plus petits" classements ne sont pas nécessairement inclus dans l'ensemble des gènes

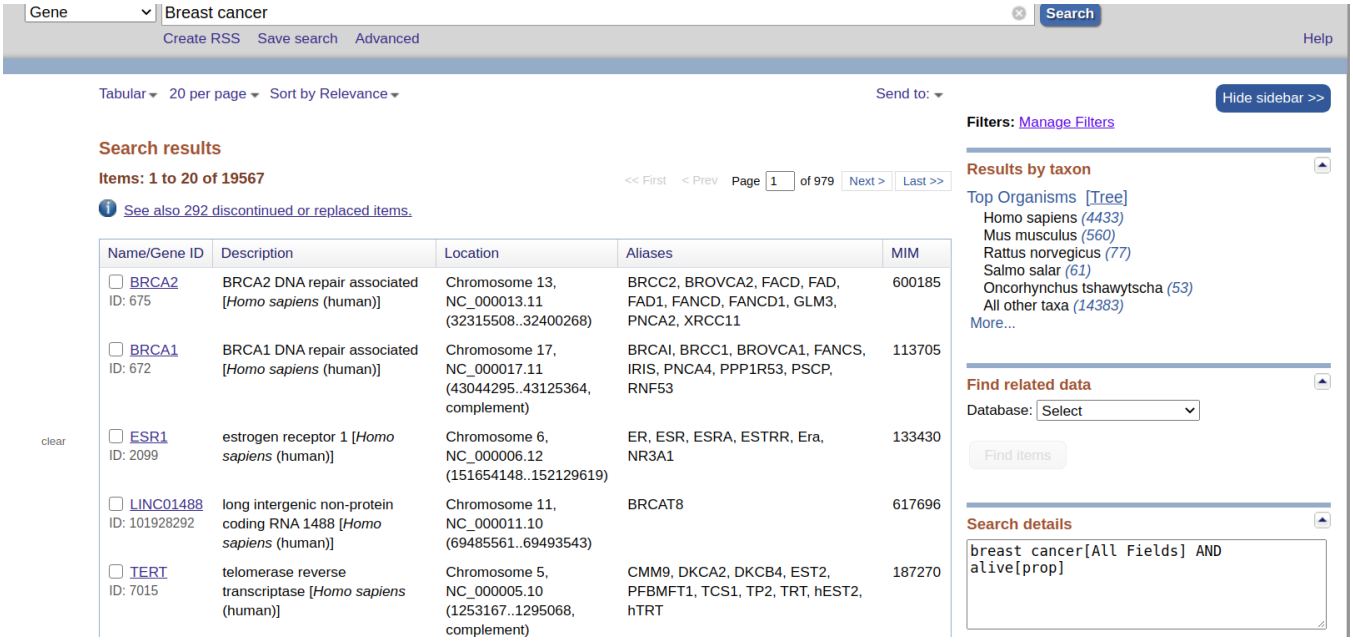


FIGURE 2 – Résultat de la requête "Breast Cancer" en interrogeant la base de données *Gene* du NCBI.

contenus dans les "plus grands classements". En revenant à l'exemple du gène *MUC1*, il est absent des résultats lorsqu'on utilise la reformulation *Cancer of the breast*.

Le problème est que les maladies peuvent avoir de nombreuses reformulations, aboutissant donc à de nombreux classements de plusieurs centaines voire milliers de gènes différents. Il est alors complexe pour les médecins de savoir quels gènes étudier en priorité. Imaginons un cas de figure caricatural mais tout à fait possible pour illustrer la difficulté de considérer plusieurs classements : le gène A est devant le gène B dans une majorité de classements, B est avant C dans une majorité de classements, et C est avant A dans une majorité de classements. Choisir un gène parmi ces trois pour une étude expérimentale s'avère être un problème complexe. L'agrégation de classements apporte des réponses à de tels problèmes.

Méthodes d'agrégation de classements. L'agrégation de classements consiste à partir de plusieurs classements en entrée et à calculer un classement dit consensuel censé représenter au mieux les points communs entre les classements d'entrée. Cette problématique intéresse depuis des siècles. Un manuscrit écrit par le majorquin Ramon Llull daté de la fin du XIIIe siècle et intitulé *'De arte eleccionis'* a été récemment découvert [33]. Ce manuscrit présente certains concepts et méthodes d'agrégation de classements considérés comme majeurs aujourd'hui.

Une méthode très utilisée dans le cadre de l'agrégation de classements est la *méthode Kemeny-Young* qui consiste à déterminer un classement aussi proche que possible des classements d'entrée, en minimisant une fonction de distance appelée *distance de Kendall-*τ. Un tel classement est appelé classement consensuel optimal ou médiane. Ce problème est connu pour être NP-difficile si le nombre de classements est pair et supérieur ou égal à 4 [20, 44] ou impair et supérieur ou égal à 7 [12]. La complexité du problème pour les cas où le nombre de

classements est égal à 3 ou 5 est encore inconnue et fait l'objet de travaux théoriques (par exemple, [70]).

Si l'agrégation de classements intéresse aujourd'hui plusieurs domaines dont l'algorithmique [2, 3, 12, 19, 20], les bases de données [46, 47, 48], la physique [73, 74], la biologie et la bioinformatique [32, 61, 64, 80], elle a été initialement étudiée dans un contexte d'élections où les électeurs étaient invités à classer l'ensemble des candidats par ordre de préférence [34, 39]. Or, les classements obtenus dans ce contexte particulier d'élections sont très différents des classements obtenus en biologie. D'une part, dans le contexte d'élections en théorie du choix social, on considère que les classements sont complets (chaque votant doit trier l'ensemble des candidats) et sans égalités (un votant ne peut pas mettre deux candidats *ex-aequo*) [9, 26, 34, 36, 39]. En conséquence, la majorité des travaux théoriques ne s'appliquent qu'à ces classements particuliers [2, 3, 19, 20, 34, 36, 37, 38, 39, 58, 67, 69, 70] et sont donc inutilisables avec des classements incomplets et/ou avec égalités comme c'est le plus souvent le cas en biologie. D'autre part, dans le contexte d'élections, on a généralement beaucoup de classements et peu d'éléments dans chaque classement (beaucoup de votants et peu de candidats). En biologie le problème est généralement orthogonal : on se retrouve avec quelques dizaines de classements dont chacun peut contenir plusieurs milliers d'entités biologiques (gènes, protéines, ...). Or, le paramètre à l'origine de la difficulté du problème d'agrégation de classements est le nombre d'éléments à trier. Ainsi, l'utilisation d'algorithmes exacts sur des données biologiques est compromise, ces derniers ne permettant pas de calculer un classement consensuel optimal en un temps raisonnable s'il y a plus de quelques dizaines d'éléments à trier. Il est donc fondamental de concevoir des heuristiques capables de calculer un classement consensuel de bonne qualité en un temps raisonnable.

Première contribution. Notre première contribution est donc de présenter une heuristique capable de gérer des données biologiques réelles (possiblement incomplètes et avec égalité). Une des étapes de cette heuristique consiste à diviser le problème initial en sous-problèmes indépendants rendant possible l'utilisation d'algorithmes exacts [7, 8]. Cette heuristique fait l'objet d'une évaluation quantitative sur un grand nombre de données biologiques massives afin d'évaluer, entre autres, sa capacité à partitionner le problème initial en de nombreux sous-problèmes [7, 8] et d'une évaluation qualitative s'intéressant à la pertinence biologique des classements consensuels calculés.

Robustesse du classement consensuel renvoyé. Un autre problème est alors soulevé : certains jeux de données peuvent avoir un très grand nombre de classements consensuels optimaux parfois très différents. Dans un tel cas, la position des éléments dans le classement consensuel, bien que ce dernier soit optimal, n'est pas robuste. Un enjeu très important dans le cadre des données réelles est ainsi de fournir à l'utilisateur des indicateurs sur la robustesse du classement consensuel renvoyé afin qu'il soit averti du degré de confiance qu'il peut avoir vis-à-vis de ce classement.

Deuxième contribution. Notre deuxième contribution est de s'attaquer au problème de la non-unicité d'un classement consensuel optimal. Plus précisément, nous présentons un algorithme capable de mettre en avant des points communs entre tous les classements consensuels optimaux. L'utilisateur peut ainsi juger la robustesse du

classement consensuel qui lui est renvoyé afin d'évaluer le niveau de confiance qu'il peut avoir vis-à-vis du résultat [7, 8]. Cet algorithme a lui aussi été évalué sur des jeux de données biologiques.

Données réelles et classements incomplets Se confronter à des données réelles implique également de se poser des questions qualitatives sur ces données. En particulier, dans le contexte d'agrégation de classements, il est légitime de se demander comment les éléments manquants dans un classement doivent être interprétés vis-à-vis de ceux présents.

Pour illustrer cette interrogation, reprenons le cas de l'agrégation de classements de gènes issus des reformulations d'une maladie puis considérons un contexte biologique différent.

Premier cas d'utilisation : reformulations synonymes de maladies. Pour rappel, les classements issus des reformulations peuvent être incomplets sans pour autant que l'ensemble des gènes contenus dans les "plus petits classements" soient inclus dans l'ensemble des gènes contenus dans les "plus grands classements". En utilisant la reformulation *Cancer of the breast*, le gène *MUC1* est absent de la liste retournée alors que le gène *MDM2* est présent. On peut en déduire que le mot clé *Cancer of the breast* n'existe pas dans la fiche de *MUC1* alors qu'il est présent dans la fiche de *MDM2*. Il paraît alors raisonnable de considérer que, vis-à-vis du mot clé *Cancer of the breast*, *MDM2* doive être considéré comme plus pertinent que *MUC1*. On préférera donc pénaliser un gène absent d'un classement, afin de ne pas mettre en avant des gènes sans aucune pertinence réelle. On retrouve cette interprétation des données manquantes dans d'autres contextes, notamment dans certains systèmes électoraux modernes où les électeurs peuvent classer jusqu'à trois candidats par ordre de préférence (c'est le cas des élections présidentielles au Sri Lanka par exemple). Dans ce contexte, il est raisonnable de considérer qu'un électeur préfère les candidats classés que les candidats non classés. Les candidats absents d'un bulletin doivent donc être pénalisés vis-à-vis des candidats présents sur le bulletin, au risque sinon de faire élire un candidat ne reflétant pas du tout les préférences des électeurs.

Deuxième cas d'utilisation : les classements issus d'expériences multi-omiques. Intéressons nous maintenant à un second cas d'utilisation possible en biologie : l'agrégation de données issues d'expériences multi-omiques. Ces expériences permettent d'observer des entités biologiques de natures différentes (gènes, ARN et protéines) mais complémentaires. Les expériences impliquant les protéines aboutissent à des classements potentiellement incomplets puisque des protocoles expérimentaux excluent certaines protéines de la mesure. Ce cas d'utilisation est ainsi très différent du précédent mettant en jeu les reformulations synonymes de maladies. En effet, dans le cas des expériences multi-omiques, les absences sont dues à un biais de protocole et ne doivent surtout pas être interprétées comme un signe de non pertinence : pénaliser l'absence peut avoir pour conséquence de passer à côté de protéines particulièrement importantes pour le phénomène biologique étudié. Les éléments présents et les éléments absents sont incomparables. Cette interprétation des données manquantes est retrouvée dans d'autres contextes, par exemple dans celui des plateformes de films que les utilisateurs peuvent noter. Même le plus ciné-

phile des utilisateurs n'aura vu qu'une petite proportion des films à disposition dans la base de données. Si un film n'a pas été noté par un utilisateur, on ne peut pas en conclure que cet utilisateur l'a trouvé pire que le film qu'il a le plus mal noté. Pénaliser les éléments absents reviendrait à favoriser les films les plus regardés au lieu de favoriser les films les plus appréciés. De la même manière, dans un contexte universitaire où des étudiants choisissent des matières optionnelles, on ne peut pas considérer que les étudiants qui n'ont pas choisi une option ont moins bien réussi l'examen que les étudiants inscrits à l'option et donc présents à l'examen.

Troisième contribution. Notre troisième contribution est de présenter un modèle dont les paramètres permettent d'intégrer un regard qualitatif sur les données, en permettant notamment de choisir la façon dont les éléments manquants dans un classement doivent être pénalisés vis-à-vis des éléments présents. Ce modèle englobe les méthodes de l'état de l'art capable de gérer les classements incomplets. Nous avons adapté les algorithmes utilisés dans le cadre des classements complets pour intégrer les paramètres du modèle, ainsi que les méthodes correspondant aux deux premières contributions. Une étude axiomatique en lien avec la théorie du choix social est également présentée, ainsi que son utilisé vis-à-vis du point de vue algorithmique. Le modèle est évalué sur des jeux de données réels de différentes natures ainsi que sur des jeux de données synthétiques.

Ce mémoire de thèse s'organise de la façon suivante :

- Le Chapitre 1 décrit formellement le problème d'agrégation de classements dans le cadre des classements complets.
- Le Chapitre 2 dresse un état de l'art des principaux algorithmes utilisés dans le cadre de l'agrégation de classements, explicite le lien entre certains de ces algorithmes et la théorie du choix social, et présente les différentes méthodes actuellement utilisées pour gérer l'agrégation de classements incomplets.
- Le Chapitre 3 présente deux algorithmes conçus pour être utilisés sur des classements incomplets et avec égalités issus de données biologiques réelles. Le premier, ParCons, est une heuristique basée sur un partitionnement du problème initial en sous-problèmes indépendants permettant dans de nombreux cas de calculer un classement consensuel optimal sur des données biologiques massives. Le deuxième, ParFront, s'attaque au problème de la non-unicité du classement consensuel optimal en mettant en évidence des points communs entre tous les classements consensuels optimaux. Ce chapitre décrit également ConQuR-BioV2, un outil en ligne destiné aux biologiques et médecins dans lequel les méthodes décrites sont implémentées. La qualité des algorithmes présentés est évaluée sur un grand nombre de jeux de données biologiques.
- Le Chapitre 4 présente un modèle paramétré permettant de gérer les classements incomplets en apportant un regard qualitatif sur les données, ainsi qu'une étude axiomatique de ce modèle inspirée de la théorie du choix social. On y explicite la manière par laquelle les algorithmes utilisés pour l'agrégation de classements complets peuvent être adaptés au modèle. Ce chapitre décrit également CoRankCo, un outil en ligne permettant de calculer des classements consensuels y compris lorsque les classements sont incomplets et avec

égalités. Le modèle est évalué sur un grand nombre de jeux de données réels (jeux de données biologiques) et synthétiques (notes d'étudiants à l'université).

- Pour terminer, nous dressons une conclusion des travaux présentés et présentons les perspectives.

Chapitre 1

L'agrégation de classements : définition du problème

Sommaire

Agréger l'information donnée par plusieurs classements est une problématique qui intéresse de nombreuses communautés. Dans cette section nous définissons dans un premier temps certains termes liés aux classements

avant de présenter formellement le problème de l'agrégation de classements au sens de la *méthode Kemeny-Young* [59, 86]. Nous commençons par le cas où les classements sont complets et sans égalités avant de présenter l'agrégation de classements complets avec égalités possibles. Le cas des classements incomplets est au coeur de nos contributions et sera développé lors des chapitres suivants.

1.1 Les classements

Dans cette section, nous présentons quelques définitions générales liées à la notion de classement.

1.1.1 Une définition générale de classement

Nous commençons par définir formellement ce qu'est un classement d'un ensemble U (U correspond à *l'univers des éléments à classer*).

Définition 1.1.1 (Classement d'un ensemble U)**.** *Un classement (ranking) r d'un ensemble U est une liste de sous-ensembles de U deux à deux disjoints appelés buckets. Plus formellement,*

$$r = [B_1, B_2, ..., B_b] \tag{1.1}$$

avec $B_i \cap B_j = \emptyset, \forall 1 \leq i \neq j \leq b$, et $\bigcup_{i=1}^{b} B_i \subseteq U$.

Exemple 1 (Exemples de classements)**.** *Soit l'ensemble $U = \{A, B, C, D\}$. Des exemples de classements de U sont présentés en Table 1.1 :*

$$\begin{aligned} r_1 &= [\{A\}, \{B\}, \{D\}, \{C\}] \\ r_2 &= [\{A, B\}, \{D\}, \{C\}] \\ r_3 &= [\{C\}, \{A\}] \\ r_4 &= [\{B, C, D\}] \end{aligned}$$

TABLE 1.1 – Exemples de classements de $U = \{A, B, C, D\}$

Par contre, $[\{A\}, \{B, E\}, \{C\}, \{D\}]$ n'est pas un classement de U car $E \notin U$. Par ailleurs, $[\{A\}, \{B\}, \{A, D\}, \{C\}]$ n'est pas un classement de U car l'élément A de U apparaît deux fois ($\{A\} \cap \{A, D\} \neq \emptyset$).

Nous définissons maintenant ce qu'est la *position* d'un élément dans un classement.

Définition 1.1.2 (Position d'un élément dans un classement)**.** *Soit $r = [B_1, B_2, ..., B_b]$ un classement de U, $x \in U$ un élément de U et i l'entier tel que $x \in B_i$. La position de x dans r, notée $r[x]$, est définit comme suit :*

$r[x] = 1 + \sum_{1 \leq j < i} |B_j|$.

Pour illustrer cette notion, nous considérons le classement r_2 de la Table 1.1. On peut voir que les éléments A et B partagent la position 1 tandis que l'élément D est en position 3.

1.1.2 Différentes caractéristiques pour les classements

Un classement peut être complet, incomplet, sans égalités, avec égalités.

Définition 1.1.3 (Classement complet, classement incomplet)**.** *Le classement r est dit complet si et seulement si $\bigcup_{i=1}^{b} B_i = U$. Dans le cas contraire, r est dit incomplet.*

Parmi les exemples de classements présentés en Table 1.1, r_1 et r_2 sont complets alors que r_3 et r_4 sont incomplets.

Définition 1.1.4 (Classement sans égalités, classement avec égalités)**.** *Le classement r est dit sans égalités si et seulement si pour tout i entre 1 et b on a $|B_i| = 1$. Dans le cas contraire, r est un classement avec égalités.*

Pour rappel, b est le nombre de *buckets*.

Parmi les exemples de classements présentés en Table 1.1, r_1 et r_3 sont sans égalités alors que r_2 et r_4 sont avec égalités.

Enfin, nous définissons le *domaine d'un classement*, noté $dom(r)$, comme l'ensemble des éléments présents dans le classement r. Une définition formelle est donnée ci-dessous.

Définition 1.1.5 (Domaine d'un classement)**.** *Le domaine d'un classement $r = [B_1, B_2, ..., B_b]$, noté $dom(r)$, est défini de la manière suivante :*

$dom(r) = \bigcup_{i=1}^{b} B_i$

Terminologies synonymes. Un classement de U complet et sans égalités est également appelé *permutation de U*. En anglais, on trouve régulièrement les dénominations *linear order* ou *total order*.

Notations. Dans l'ensemble de cette thèse, U désignera l'ensemble des éléments à trier. Selon le contexte, les éléments de U peuvent désigner par exemple des gènes, des candidats à une élection, des films.

Pour un ensemble U, on notera $P(U)$ l'ensemble des permutations de U, $C(U)$ l'ensemble des classements complets de U (avec ou sans égalités), $Z(U)$ l'ensemble des classements sans égalités de U, $A(U)$ l'ensemble des classements de U (complets ou incomplets, avec ou sans égalités).

Un classement complet, en anglais *ranking with ties*, est parfois appelé *weak order* ou *partial ranking* [47]. Ce dernier terme est ambigu et peut également faire référence aux classements incomplets (par exemple dans [44]).

Remarque sur le nombre de classements complets d'un ensemble U. Étant donné un ensemble U de taille n, le nombre de permutations de U est égal à $n!$. Le nombre de classements complets avec ou sans égalités de U correspond au $n - ieme$ nombre de Fubini (également appelé $n - ieme$ nombre de Bell ordonné) et a pour valeur approchée $\frac{n!}{2*(ln(2))^{n+1}}$.

1.1.3 Positions relatives des éléments dans un classement

Définition 1.1.6 (Avant, Après, À égalité)**.** *Soient r un classement et $x \neq y$ deux éléments appartenant au domaine de r. On dira que x est avant y dans r, noté $x \prec_r y$, ssi $r[x] < r[y]$. De la même manière, on dira que x est après y dans r, noté $x \succ_r y$, ssi y est avant x dans r. Enfin, on dira que x est à égalité avec y dans r, noté $x \equiv_r y$, ssi $r[x] = r[y]$. Dans le cas contraire, on notera $x \not\equiv_r y$.*

1.2 Agrégation de permutations selon la méthode Kemeny-Young

Plusieurs méthodes ont été décrites dans le cadre de l'agrégation de permutations [36, 39, 59, 79]. Dans cette thèse, nous nous intéresserons particulièrement à la méthode Kemeny-Young visant à minimiser une mesure de distance.

1.2.1 Distance de Kendall-τ

L'objectif de l'agrégation de classements est de calculer un classement dit consensuel qui soit le plus proche possible de la liste de classements que l'on cherche à agréger. La distance de Kendall-τ est une fonction intéressante puisqu'elle prend en entrée deux permutations et renvoie une distance entre ces deux permutations. Elle correspond au nombre de paires d'éléments dont l'ordre relatif est inversé entre les deux permutations. Sa définition formelle est donnée ci-dessous.

Définition 1.2.1 (Distance de Kendall-τ)**.** *Soient r_1 et r_2 deux permutations d'un ensemble U. La distance de Kendall-τ entre r_1 et r_2, notée $d(r_1, r_2)$, est définie de la manière suivante :*

$$d(r_1, r_2) = |\{(x, y) \in U^2 : x \prec_{r_1} y \wedge y \prec_{r_2} x\}| \tag{1.2}$$

Cette distance a pour valeur 0 si et seulement si r_1 et r_2 sont identiques. La valeur maximale est atteinte si et seulement si les deux permutations sont miroir l'une de l'autre (par exemple $[\{A\}, \{B\}, \{C\}]$ et $[\{C\}, \{B\}, \{A\}]$). Dans ce cas, la distance est égale au nombre de paires d'éléments distincts de U à savoir $\frac{|U|*(|U|-1)}{2}$.

Remarque sur la distance de Kendall-τ : Si on voit les deux permutations comme deux tableaux d'entiers, la distance de Kendall-τ est égale au nombre d'échanges que ferait l'algorithme du tri à bulles pour passer du premier au deuxième tableau. Ainsi, cette distance est également appelée en anglais *bubble sort distance*. Par ailleurs, si les éléments de U sont renommés de sorte que la première permutation devienne $[\{1\}, \{2\}, \ldots, \{|U|\}]$, cette distance peut être également vue comme le nombre d'inversions dans la deuxième permutation. Compter les inversions est un problème bien connu dans la littérature. L'algorithme naïf consistant à regarder chaque paire d'éléments a une complexité $\Theta(|U|^2)$. Une bien meilleure version de complexité $\Theta(|U| * \log |U|)$ inspirée du tri fusion

est couramment utilisée [44]. Le meilleur algorithme à ce jour (plus complexe et très peu utilisé en pratique) a une complexité $\Theta(|U| * \sqrt{\log |U|})$ [29].

Exemple 2 (Calcul de distance de Kendall-τ)**.** *Soit une première permutation* $r_1 = [\{A\}, \{C\}, \{B\}, \{D\}]$ *et une deuxième permutation* $r_2 = [\{B\}, \{C\}, \{A\}, \{D\}]$*. Prenons l'exemple de la paire* (A, B)*. On voit que* A *est avant* B *dans* r_1 *alors que* A *est après* B *dans* r_2*. Cette paire induit donc une pénalité de 1. En revanche, si on regarde la paire* (A, D)*, on voit que* A *est avant* D *dans* r_1 *et dans* r_2*. Cette paire n'induit donc aucune pénalité. Au final, en répétant ce processus pour toutes les paires d'éléments de* $U = \{A, B, C, D\}$*, on obtient :*

$d(r_1, r_2) = 1_{(A,B)} + 1_{(A,C)} + 0_{(A,D)} + 1_{(B,C)} + 0_{(B,D)} + 0_{(C,D)} = 3.$

1.2.2 Le score de Kemeny

Le score de Kemeny est une mesure permettant d'évaluer à quel point une permutation r est représentative d'une liste [1] de permutations R. Cette mesure correspond à la somme des distances de Kendall-τ entre r et chaque permutation dans R.

Définition 1.2.2 (Score de Kemeny)**.** *Le score de Kemeny entre une permutation* r *et une liste de permutations* R*, noté* $S(r, R)$*, est défini comme suit :*

$$S(r, R) = \sum_{r_i \in R} d(r, r_i) \tag{1.3}$$

Exemple 3 (Calcul du score de Kemeny)**.** *On considère pour cet exemple une liste* $R = [r_1, r_2, r_3, r_4]$ *constituée des quatre permutations suivantes :*

$r_1 = [\{A\}, \{C\}, \{D\}, \{B\}]$

$r_2 = [\{B\}, \{A\}, \{C\}, \{D\}]$

$r_3 = [\{A\}, \{B\}, \{D\}, \{C\}]$

$r_4 = [\{B\}, \{D\}, \{C\}, \{A\}]$

On considère maintenant la permutation suivante : $r = [\{B\}, \{A\}, \{D\}, \{C\}]$ *(faisant office de classement consensuel possible pour* R*) . En utilisant la définition du score de Kemeny, on obtient :*

$S(r, R) = d(r, r_1) + d(r, r_2) + d(r, r_3) + d(r, r_4) = 4 + 1 + 1 + 2 = 8.$

Intérêt du score de Kemeny.

Le score de Kemeny permet d'évaluer à quel point une permutation r est éloignée d'une liste de permutations R. Ainsi, plus le score de Kemeny est bas, mieux la permutation r représente R.

1. On préfère utiliser une liste plutôt qu'un ensemble, car l'ordre des permutations peut avoir de l'importance pour certaines méthodes d'agrégation de classements que l'on trouve en théorie du choix social. Une méthode d'agrégation de classements dont l'ordre des permutations n'a pas d'importance est dite *anonyme* [26].

Revenons à l'Exemple 3 et considérons maintenant une deuxième permutation $s = [\{C\}, \{D\}, \{A\}, \{B\}]$. La question est maintenant de savoir laquelle des permutations r ou s représente le mieux la liste R. On a $S(r, R) = 8$ et $S(s, R) = 2 + 5 + 4 + 4 = 15$. Comme $S(r, R) < S(s, R)$, on peut en conclure que r est une permutation plus représentative de R que s.

1.2.3 Classement consensuel optimal

Un *classement consensuel optimal*, également appelé *médiane*, est une permutation r^* minimisant le score de Kemeny. Une définition formelle est donnée ci-dessous.

Définition 1.2.3 (Classement consensuel optimal)**.** *Soit U un ensemble et R une liste de permutations de U. Un classement consensuel optimal, noté r^*, est une permutation de U respectant la propriété suivante :*

$$\forall r \in P(U), S(r^*, R) \leq S(r, R) \tag{1.4}$$

Remarque sur la non-unicité d'un classement consensuel optimal. Il est important de noter qu'il peut y avoir plusieurs classements consensuels optimaux pour une même liste de permutations. Dans le contexte de l'agrégation de permutations, il peut y avoir autant de classements consensuels optimaux que de permutations de U à savoir $|U|!$ [1].

Exemple 4 (Exemples de classements consensuels optimaux)**.** *Les classements consensuels optimaux pour la liste de classements R de l'Exemple 3 sont au nombre de quatre :*

$r_1^* = [\{A\}, \{B\}, \{C\}, \{D\}]$*,* $r_2^* = [\{A\}, \{B\}, \{D\}, \{C\}]$*,* $r_3^* = [\{B\}, \{A\}, \{C\}, \{D\}]$ *et* $r_4^* = [\{B\}, \{A\}, \{D\}, \{C\}]$*.*

Le score de Kemeny associé est de 8.

Terminologies synonymes. Le classement consensuel optimal est également appelé en anglais *optimal Kemeny ranking* [2, 5, 14, 19], *optimal aggregation* [44], *optimal solution* [3, 55, 67, 78] ou encore médiane (median ranking) [32].

1.2.4 Complexité du problème

Savoir s'il existe un classement consensuel dont le score de Kemeny est inférieur ou égal à un entier k donné est un problème NP-difficile lorsque le nombre de permutations dans la liste R est pair et supérieur ou égal à 4 et que le nombre d'éléments de l'ensemble U est supérieur ou égal à 4 [20, 44]. Ce problème est facile si le nombre de permutations est de 2. Une publication récente montre que le problème d'agrégation de classements est également NP-difficile lorsque le nombre de permutations est supérieur à 7 [12]. La complexité du problème dans le cas où le

1. C'est le cas si, par exemple, $R = [r_1, r_2]$ avec r_1 et r_2 deux permutations miroir l'une de l'autre.

nombre de permutations est égal à 3 ou 5 est encore ouvert. Des heuristiques et algorithmes d'approximation sont présentés dans le Chapitre 2.

1.3 Adaptation pour les classements avec égalités

Cette section présente une généralisation de la méthode Kemeny-Young définie par Fagin *et al.* [47] permettant que les classements d'entrée contiennent des égalités.

1.3.1 Distance de Kendall-τ généralisée avec pénalité p

Fagin et al. ont généralisé la distance de Kendall-τ pour permettre d'obtenir une distance entre deux classements complets pouvant contenir des égalités. Cette fonction de distance possède un paramètre réel noté p correspondant au coût de création ou de rupture d'une égalité.

Définition 1.3.1 (Distance de Kendall-τ généralisée avec pénalité p)**.** *La distance de Kendall-τ au généralisée avec pénalité p entre deux classements r_1 et r_2 complets de U, notée $G^p(r_1, r_2)$ avec $p \in]0, 1]$ est définie de la manière suivante :*

$$G^p(r_1, r_2) = |\{(x, y) \in U^2 : x \prec_{r_1} y \wedge y \prec_{r_2} x\}| + p * |\{(x, y) \in U^2 : x \equiv_{r_1} y \wedge x \prec_{r_2} y \vee x \prec_{r_1} y \wedge x \equiv_{r_2} y\}| \quad (1.5)$$

Ainsi, pour une paire d'éléments distincts (x, y) de U et deux classements r_1 et r_2, il y a trois coûts possibles :

- 1 si $x \prec_{r_1} y$ et $y \prec_{r_2} x$ ou alors si $y \prec_{r_1} x$ et $x \prec_{r_2} y$.
- p si $x \prec_{r_1} y$ et $x \prec_{r_2} y$ ou alors si $x \not\equiv_{r_1} y$ et $x \equiv_{r_2} y$.
- 0 pour tous les autres cas.

Exemple 5 (Calcul de distance de Kendall-τ généralisée)**.** *Soit $U = \{A, B, C, D\}$, et deux classements complets $r_1 = [\{A, B\}, \{C\}, \{D\}]$, $r_2 = [\{B\}, \{C\}, \{A\}, \{D\}]$. Prenons l'exemple de la paire (A, B). On voit que A est à égalité avec B dans r_1 alors que A est après B dans r_2. Cette paire induit donc une pénalité de p. Si on regarde la paire (A, D), on voit que A est avant D dans r_1 et dans r_2. Cette paire n'induit donc aucune pénalité. Enfin, en regardant la paire (A, C), on voit que A est avant C dans r_1 alors que A est après C dans r_2. Cette paire induit une pénalité de 1. Au final, en répétant ce processus pour toutes les paires de U, on obtient :*

$G^p(r_1, r_2) = p_{(A,B)} + 1_{(A,C)} + 0_{(A,D)} + 0_{(B,C)} + 0_{(B,D)} + 0_{(C,D)} = 1 + p$. *En particulier, on a $G^1(r_1, r_2) = 2$ et $G^{0.5}(r_1, r_2) = 1.5$.*

1.3.2 Score de Kemeny généralisé avec pénalité p

Sur le même principe que le score de Kemeny, étant donné un classement complet r et une liste de classements complets R, le score de Kemeny généralisé avec pénalité p correspond à la somme des distances de Kendall-τ généralisée entre r et chaque classement de R.

Définition 1.3.2 (Score de Kemeny généralisé avec pénalité p)**.** *Le score de Kemeny généralisé avec pénalité p entre un classement complet r et une liste de classements complets R, noté $S^p(r, R)$ avec $p \in]0, 1]$, est défini de la manière suivante :*

$$S^p(r, R) = \sum_{r_i \in R} G^p(r, r_i) \tag{1.6}$$

Comme le score de Kemeny, le score de Kemeny généralisé avec pénalité p permet d'évaluer à quel point un classement complet unique (classement consensuel) représente bien une liste de classements.

Exemple 6 (Calcul de score de Kemeny généralisé)**.** *On considère pour cet exemple la liste $R = [r_1, r_2, r_3, r_4]$ constitué des quatre classements complets suivants :*

$r_1 = [\{A\}, \{B, C\, D\}]$

$r_2 = [\{B\}, \{A, C\}, \{D\}]$

$r_3 = [\{A\}, \{B\}, \{C, D\}]$

$r_4 = [\{B\}, \{D\}, \{A, C\}]$

On considère maintenant le classement complet suivant : $r = [\{B\}, \{A, D\}, \{C\}]$. En utilisant la définition du score de Kemeny généralisé avec pénalité p, on obtient :

$S^p(r, R) = G^p(r, r_1) + G^p(r, r_2) + G^p(r, r_3) + G^p(r, r_4) = (1 + 4 * p) + (1 + 2 * p) + (1 + 2 * p) + 2p = 3 + 10 * p.$

1.3.3 Classement consensuel optimal

Un *classement consensuel optimal*, également appelé *médiane*, est un classement complet r^* minimisant le score de Kemeny. Plus formellement, nous définissons un classement consensuel optimal de la manière suivante :

Définition 1.3.3 (Classement consensuel optimal)**.** *Soit U un ensemble, R une liste de classements complets de U et $p \in]0, 1]$. Un classement consensuel optimal, noté r^*, est défini de la manière suivante :*

$$\forall r \in C(U), S^p(r^*, R) \leq S^p(r, R) \tag{1.7}$$

A nouveau, il est important de noter qu'il peut y avoir plusieurs classements consensuels optimaux pour une même liste de classements. On peut aisément construire des jeux de données pour lesquels chaque classement

complet de U est un classement consensuel optimal pour R[1].

Exemple 7 (Exemple de classements consensuels optimaux)**.** *En prenant $p = 1$, les classements consensuels optimaux pour la liste de classements R de l'Exemple 6 sont au nombre de deux :*

$r_1^* = [\{A\}, \{B\}, \{C, D\}]$ *et* $r_2^* = [\{B\}, \{A\}, \{C, D\}]$*. Le score associé est de 9.*

En prenant $p = 0.5$, les classements consensuels optimaux pour la liste de classements R de l'Exemple 6 sont au nombre de trois :

$r_1^* = [\{A\}, \{B\}, \{C, D\}]$*,* $r_2^* = [\{B\}, \{A\}, \{C, D\}]$ *et* $r_3^* = [\{A, B\}, \{C, D\}]$*. Le score associé est de 8.*

1.3.4 Complexité de l'agrégation de classements avec égalités

L'agrégation de classements avec égalités est un problème NP-difficile lorsque $p = 1$ [23]. La démonstration tient dans le fait qu'avec $p = 1$, les classements consensuels optimaux d'une liste de permutations sont nécessairement des permutations (on revient au problème d'agrégation de permutations).

1.3.5 D'autres généralisations aux classements complets

La généralisation de la méthode Kemeny-Young pour les classements complets avec égalités présentée dans [53] est équivalente à un facteur 2 près à la distance de Kendall-τ généralisée avec pénalité p de Fagin *et al.* en prenant $p = 0.5$. En effet, dans [53], le coût de créer/rompre une égalité a un coût de 1 tandis qu'une inversion a un coût de 2. Enfin, Truchon a également proposé une manière de considérer les égalités dans les classements d'entrée lorsque l'on veut un classement consensuel sans égalités [83]. La méthode proposée revient à utiliser la distance de Kendall-τ généralisée avec pénalité p de Fagin *et al.* en prenant $p = 1$ mais en contraignant que r (le classement consensuel) soit sans égalités.

1.4 Conclusion

Dans ce premier chapitre, nous avons posé les bases pour nous permettre de comprendre le problème d'agrégation de classements. Nous avons commencé par donner une définition formelle au terme "classement" suffisamment générique pour couvrir les concepts de classement avec ou sans égalités ainsi que classement complet ou incomplet.

Nous avons alors défini le problème d'agrégation de classements tel qu'il a été formalisé initialement, c'est-à-dire dans le cas où les classements dont on cherche un consensus sont complets et sans égalités (permutations).

1. Avec par exemple $p = 1$ et $U = \{A, B, C\}$, il suffit de prendre $R = [r_1, r_2, r_3]$ avec $r_1 = [\{A\}, \{B\}, \{C\}]$, $r_2 = [\{C\}, \{B\}, \{A\}]$ et $r_3 = [\{A, B, C\}]$

Cette version du problème est celle que l'on rencontre dans la majorité des travaux publiés sur l'agrégation de classements et que l'on utilise en particulier dans la communauté théorie du choix social.

Nous avons ensuite présenté la généralisation du problème aux classements avec égalités. Cette généralisation introduit un paramètre réel $p \in]0, 1]$ qui correspond au coût de création/rupture d'égalité. On peut s'interroger sur les conséquences du choix de la valeur de p sur le classement consensuel calculé lorsque l'on considère des données réelles. Des éléments de réponse seront donnés dans le Chapitre 4.

Résumé du Chapitre 1

Ce chapitre nous a permis de poser formellement le problème d'agrégation de classements. Nous avons commencé par définir un classement comme une liste de sous-ensembles $[B_1, B_2, \ldots, B_b]$ (appelés *buckets*) deux à deux disjoints d'un ensemble U de départ qui correspond aux éléments à classer.

Nous avons donné une définition aux concepts de classement complet (l'union des sous-ensembles B_i est égale à U), classement incomplet (l'union des sous-ensembles B_i est strictement incluse dans U), classement avec égalités (au moins un *bucket* est de taille strictement supérieure à 1) et classement sans égalités (tous les *buckets* sont de taille 1) qui seront utilisés tout au long de ce mémoire de thèse. Par souci de simplification, lorsque les classements sont complets et sans égalités, on parlera de permutation.

Deux éléments sont dits à égalité (*ex aequo*) s'ils sont dans le même *bucket*. Un élément x est avant un élément y si le *bucket* contenant x est avant le *bucket* contenant y dans la liste.

La position d'un élément x vaut 1 + le nombre d'éléments positionnés avant x dans le classement.

La distance de Kendall-τ permet de mesurer la distance entre deux permutations de U. Sa valeur correspond au nombre de paires d'éléments de U dont l'ordre relatif est inversé entre les deux permutations. Pour rappel, l'ordre relatif de deux éléments x et y est inversé entre deux permutations r_1 et r_2 si x est avant (resp. après) y dans r_1 alors que x est après (resp. avant) y dans r_2.

Dans sa version classique, le problème d'agrégation de classements consiste à partir d'une liste de permutations de U et à déterminer, parmi l'ensemble de toutes les permutations de U, celle(s) qui minimise(nt) le score de Kemeny défini comme la somme des distances de Kendall-τ avec les permutations de départ. Ces permutations sont alors appelées classements consensuels optimaux ou médianes. Le problème est connu pour être NP-difficile dans la majorité des cas [20, 44]. Le paramètre à l'origine de la difficulté de ce problème n'est pas le nombre de permutations mais le nombre d'éléments à trier (le nombre d'éléments de U).

Le problème d'agrégation de classements a été généralisé aux classements complets avec égalités par Fagin *et al.* [47]. Ces derniers ont défini la distance de Kendall-τ généralisée avec paramètre p. Cette distance mesure l'éloignement entre deux classements complets avec ou sans égalités. La valeur de p correspond au coût de création ou de rupture d'égalité : il s'agit de la pénalité lorsque deux éléments sont à égalité dans un seul des deux classements.

Dans ce contexte, le problème d'agrégation de classements consiste à partir d'une liste de classements complets (avec ou sans égalités) et de déterminer parmi l'ensemble des classements complets de U celui ou ceux qui minimise(nt) le score de Kemeny généralisé avec pénalité p (défini comme la somme des distances de Kendall-τ généralisée avec les classements complets de départ). Le problème est au moins aussi difficile que dans le cas des permutations pour une valeur de p égale à 1 [23].

Chapitre 2

L'agrégation de classements : État de l'art

Sommaire

Le problème d'agrégation de classements a été essentiellement étudié sous deux aspects : l'aspect axiomatique qui intéresse particulièrement la théorie du choix social [9, 10, 26] et l'aspect algorithmique [3, 17, 44]. Ces deux aspects sont très complémentaires et ces deux communautés s'inspirent l'une de l'autre. D'un côté on trouve des ouvrages de référence en théorie du choix social avec une étude poussée sur les aspects algorithmiques et combinatoires [9, 26]. De l'autre côté, certains articles de la communauté algorithmique s'intéressent à des problèmes de complexité récurrents en théorie du choix social ou [16] ou utilisent des *critères d'équité* (*fairness criteria*) de la théorie du choix social pour concevoir des algorithmes [19, 78]. Les critères d'équité permettent de caractériser

les méthodes d'agrégation de classements. Un exemple de critère est le *critère de majorité* qui stipule que si un candidat (élément de U) est premier dans une stricte majorité de classements, alors il doit remporter l'élection.

Calculer un classement consensuel optimal (au sens de minimiser le score de Kemeny) est un problème NP-difficile dans de nombreux cas [12, 20, 44], ce qui explique l'intérêt réciproque des communautés théorie du choix social et algorithmique. Le paramètre qui rend le problème difficile est le nombre d'éléments à classer. Par contre, le problème est polynomial en fonction du nombre de classements [12, 20, 44]. Un pan de la recherche algorithmique consiste à concevoir des algorithmes exacts aussi rapides que possible [5, 11, 30, 35, 38, 69, 78, 84]. En pratique, ces algorithmes exacts permettent de calculer un classement consensuel optimal lorsque le nombre d'éléments à classer est de l'ordre de quelques dizaines. Dans un contexte de système électoral en théorie du choix social, ces algorithmes sont utilisables puisque le nombre de candidats à classer est généralement très petit par rapport au très grand nombre d'électeurs qui classent ces candidats. En revanche, dans d'autres contextes comme l'interrogation de bases de données biologiques, on se retrouve avec le problème orthogonal c'est-à-dire peu de classements mais plusieurs milliers d'éléments à classer. Dans un tel cas, les algorithmes exacts sont impossibles à utiliser. Afin de réduire leur temps d'exécution, des techniques de réduction d'espace ont été proposées [17, 83]. Elles consistent le plus souvent à diviser le problème initial en sous-problèmes indépendants. On trouve aussi des techniques de réduction d'espace qui consistent à déterminer des ordres relatifs entre certains éléments dans les classements consensuels optimaux sans pour autant aboutir à un partitionnement [68, 69]. Ces techniques de réduction d'espace peuvent néanmoins s'avérer inefficaces sur de nombreuses instances pour lesquelles il est très important de pouvoir calculer rapidement un classement consensuel de bonne qualité, aussi proche que possible d'un classement consensuel optimal. Ainsi, un enjeu majeur est de concevoir des algorithmes d'approximation et des heuristiques fiables permettant de renvoyer un classement consensuel de bonne qualité en un temps raisonnable. De nombreux algorithmes d'approximation et heuristiques sont classiquement utilisées dans le cas où les classements sont complets et sans égalités [2, 3, 36, 39, 46].

Si calculer un classement consensuel optimal est un enjeu important, le simple critère d'optimalité au sens du score de Kemeny ne suffit pas à garantir la représentativité d'un classement consensuel vis-à-vis d'une liste de classements (à quel point un classement consensuel permet de mieux rendre compte des points communs et différentes entre les classements d'entrée qu'un autre classement consensuel). En effet, même lorsqu'un jeu de données est suffisamment petit pour qu'un algorithme exact soit utilisé, le nombre de classements consensuels optimaux renvoyé peut être gigantesque. Les auteurs de [74] calculent l'ensemble des classements consensuels optimaux qu'ils transforment en un classement consensuel unique en regardant la position de chaque élément dans chaque classement consensuel optimal. Cette méthode n'est pas applicable sur des jeux de données biologiques contenant plusieurs centaines voire milliers d'éléments en raison du temps d'exécution nécessaire pour une telle approche. Un enjeu est donc de trouver des points communs entre les différentes solutions optimales sans les calculer de façon exhaustive.

Enfin, un autre problème important à considérer est la gestion des classements incomplets avec égalités. Pour la gestion des égalités, le modèle de Fagin *et al.* [47] (voir Section 1.3) fait consensus. A l'inverse, il n'y a pas de consensus dans la littérature quant à une "bonne" façon de généraliser la méthode Kemeny-Young aux classements incomplets. En conséquence, de nombreuses méthodes co-existent [4, 17, 19, 24, 44, 78] sans pour autant que leurs cas d'utilisation sur des données réelles soient explicités. Il est donc très difficile de savoir dans quels cas une généralisation donnée est plus pertinente qu'une autre. De plus, on ne trouve ni algorithme exact ni technique de réduction d'espace dans le cas où les classements sont incomplets avec égalités. Les heuristiques proposées comme BioConsert [24] et GRASP [4] n'offrent aucune garantie quant à la qualité du classement consensuel renvoyé et ne peuvent servir que pour une généralisation donnée du score de Kemeny (celle publiée avec l'heuristique).

Dans la Section 2.1, nous commencerons par présenter l'agrégation de classements à travers un prisme algorithmique en présentant les algorithmes exacts ainsi que les algorithmes d'approximation et heuristiques les plus couramment utilisés dans ce cadre. La Section 2.2 décrit des méthodes de partitionnement permettant de diviser le problème initial en sous-problèmes indépendants ainsi que les liens étroits entre ces techniques de partitionnement et la théorie du choix social. Dans la Section 2.3, nous présenterons les différentes méthodes utilisées dans la littérature pour gérer les classements incomplets. Nous conclurons cet état de l'art dans la Section 2.4.

2.1 Algorithmique pour l'agrégation de classements

De nombreux algorithmes existent pour répondre au problème de l'agrégation de classements. Ces algorithmes sont majoritairement conçus pour gérer les permutations mais certains peuvent parfois être adaptés pour une utilisation dans un spectre plus large. Nous verrons d'abord qu'une représentation des classements d'entrée par un graphe permet de faire le pont entre le problème d'agrégation de classements et un autre problème algorithmique récurrent en théorie des graphes : le *weighted feedback arc set problem*. Nous présenterons ensuite des algorithmes exacts pour l'agrégation de permutations, puis nous ferons le point sur des techniques de réduction d'espace utiles pour réduire le temps nécessaire aux algorithmes exacts. Nous discuterons ensuite du lien étroit entre les techniques de réduction d'espace et deux particulièrement importants de la théorie du choix social. Enfin, nous présenterons certaines heuristiques dont l'intérêt reste entier lorsque, même couplés aux techniques de réduction d'espace, les algorithmes exacts ne permettent pas de calculer un classement consensuel en un temps raisonnable.

2.1.1 Représentation des permutations à l'aide de graphes

L'utilisation d'algorithmes à base de graphes est classique dans le cadre de l'agrégation de permutations [44, 58] (classements complets sans égalités). L'idée générale est de partir d'une liste de permutations et de passer du

problème d'agrégation de permutations à un problème de graphes bien connu. Pour ce faire, il faut construire le graphe orienté pondéré défini ci-dessous.

Définition 2.1.1 ([58])**.** *Soit $G = (V, A)$ le graphe orienté pondéré tel que :*

- *$V = U$ (autrement dit les sommets sont les éléments à classer).*
- *$(x, y) \in A$ si et seulement si l'élément x est avant l'élément y dans une stricte majorité de permutations.*
- *le poids de $(x, y) \in A$ correspond à la différence entre le nombre de permutations telles que x est avant y et le nombre de permutations telles que y est avant x.*

Remarque. Intuitivement, on peut voir le poids d'un arc comme le nombre minimal de classements qu'il faudrait ajouter pour faire disparaître cet arc. Afin d'illustrer le lien entre permutations et graphes, nous présentons une liste de 3 permutations de 4 éléments en Table 2.1 ainsi que le graphe associé en Figure 2.1.

$$r_1 = [\{A\}, \{B\}, \{C\}, \{D\}]$$
$$r_2 = [\{A\}, \{C\}, \{D\}, \{B\}]$$
$$r_3 = [\{A\}, \{D\}, \{B\}, \{C\}]$$

TABLE 2.1 – Exemple de liste de 3 permutations de 4 éléments

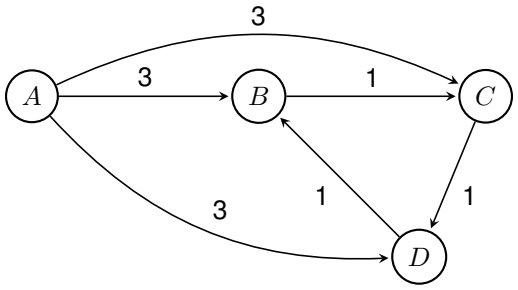


FIGURE 2.1 – Graphe représentant les permutations de la Table 2.1

Le graphe obtenu (Figure 2.1) comporte un cycle formé par les éléments B, C, D. L'existence de ce cycle est liée à la possibilité d'avoir à la fois B avant C, C avant D et D avant B dans une stricte majorité de classements. Or, il est impossible de construire un classement consensuel plaçant à la fois B avant C, C avant D et D avant B. Trouver un classement consensuel optimal revient à rendre le graphe acyclique en retirant un ensemble d'arcs dont la somme des poids est minimale. En effet, retirer un arc permet de résoudre des conflits, forçant ainsi un ordre. Le prix à payer pour retirer un arc correspond au poids de l'arc. Par exemple, retirer l'arc de D vers B dans le graphe présenté en Figure 2.1 permet de de supprimer le cycle formé par A, B, C rendant l'ensemble du graphe acyclique. Cette opération a un coût de 1 ce qui correspond au poids de l'arc (D, B).

Ce problème correspond au très célèbre *weighted feedback arc set problem* [44, 59, 60]. Ainsi, tous les travaux en théorie des graphes concernant le *weighted feedback arc set problem* sont directement utilisables dans le cadre

de l'agrégation de permutations. Claire Kenyon-Mathieu *et al.* [60] ont par exemple présenté plusieurs schémas d'approximation en temps polynomial pour le *weighted feedback arcset problem* [60]. Dans [35], les graphes sont utilisés pour borner le score de Kemeny minimal. Par ailleurs, un algorithme à base de graphe a également été proposé par Betzler et al. [19] et sera discuté dans la Section 2.2.

Malheureusement, le graphe G présenté dans la Définition 2.1.1 ne permet de considérer ni les classements avec égalités, ni les classements incomplets. La prise en compte des égalités nécessite en effet de redéfinir l'ensemble des arcs du graphe afin de pouvoir représenter le cas où l'ordre relatif le moins coûteux pour une paire (x, y) dans un classement consensuel serait l'égalité. Considérons dans un premier temps une liste R de permutations de U et deux éléments x et y de U. On peut se demander s'il est plus intéressant de placer x avant y ou y avant x dans le classement consensuel (indépendamment des autres éléments). Pour ce faire, comparons le nombre de permutations plaçant x avant y, qu'on appelle ici k_1, et le nombre de permutations plaçant y avant x, qu'on appelle ici k_2. L'idée est donc de regarder quelle(s) valeur(s) est/sont minimale(s) parmi k_1 et k_2.

Puisque les classements sont ici sont des permutations, il n'y a que trois possibilités :

- $k1 > k2$: dans ce cas on place un arc de x vers y avec un poids de $k1 - k2$.
- $k1 < k2$: dans ce cas on place un arc de y vers x avec un poids de $k2 - k1$.
- $k1 = k2$: dans ce cas on ne place aucun arc entre x et y.

Considérons maintenant une liste de classements complets avec égalités R' et considérons une paire (x, y) d'éléments. Appelons respectivement $k1$, $k2$ et $k3$ le nombre de classements de R' tels que x est avant y, y est avant x et x est à égalité avec y dans les classements de R'. L'objectif est à nouveau de regarder quelle(s) valeur(s) est/sont minimale(s) parmi $k1$, $k2$ et $k3$. Dans ce cas, on ne dénombre pas moins de 7 possibilités (les valeurs minimales ont été mises en gras) :

- **k1** $< k2 \leq k3$
- **k2** $< k1 \leq k3$
- **k3** $< k1 \leq k2$
- **k1** $=$ **k2** $< k3$
- **k2** $=$ **k3** $< k1$
- **k1** $=$ **k3** $< k2$
- **k1** $=$ **k2** $=$ **k3**

Généraliser le graphe présenté en Définition 2.1.1 pour les classements avec égalités n'est pas trivial étant donné le nombre de nouveaux cas à prendre en compte. Nous apportons un élément de réponse à ce problème au Chapitre 3.

De la même manière, si les classements de R' peuvent également être incomplets, il faut alors s'intéresser au nombre de classements tels que x est présent et y est absent, au nombre de classements tels que y est présent et x est absent, et au nombre de classements tels que x et y sont absents. Il devient difficile pour un graphe de faire

ressortir l'ensemble de ces informations pourtant primordiales.

2.1.2 Algorithmes exacts

Nous avons vu dans la Sous-section 2.1.1 que représenter les permutations à l'aide du graphe présenté précédemment (voir Définition 2.1.1) permet l'utilisation dans le contexte d'agrégation de classements d'algorithmes initialement conçus pour répondre au *weighted feedback arcset problem* (exacts ou non). Nous allons maintenant présenter des algorithmes exacts conçus spécifiquement pour le problème d'agrégation de classements. Deux approches sont très majoritaires dans ce cadre : la programmation linéaire en nombre entiers et les algorithmes de type séparation et évaluation.

Programmation linéaire en nombre entiers (PLNE)

Une manière classique et naturelle de concevoir un algorithme exact pour calculer un classement consensuel optimal (ou plusieurs classements consensuels optimaux) est la programmation linéaire en nombres entiers. Dans le cas du problème d'agrégation de permutations, la formulation présentée à plusieurs reprises dans la littérature [5, 23, 68, 78] est rappelée en Table 2.2.

minimize	$\sum_{x \neq y \in U} Q_{x \prec y} * b_{y,x} + Q_{y \prec x} * b_{x,y}$	(1)
subject to	$b_{x,y} \in \{0,1\}, \forall x \neq y \in U$	(2)
	$b_{x,y} + b_{y,x} = 1$	(3)
	$b_{x,y} \leq b_{x,z} + b_{z,y}, \forall x \neq y \neq z$	(4)

TABLE 2.2 – Algorithme exact en PLNE pour le problème d'agrégation de permutations.

La ligne (1) correspond à la fonction objectif à minimiser. Pour chaque paire d'éléments (x, y), on définit $b_{x,y}$ comme le booléen qui vaut 1 si et seulement si x est avant y dans le classement consensuel et $Q_{x \prec y}$ comme le nombre de fois où x est avant y dans les classements d'entrée. La fonction objectif se décompose comme une somme sur les paires (x, y) d'éléments du coût induit par le positionnement relatif de x et y dans le classement consensuel considéré. Si dans le classement consensuel x est avant y (respectivement y est avant x), le coût induit par cette paire correspond au nombre de classements en désaccord avec cet ordre à savoir le nombre de classements tels que y est avant x (respectivement x est avant y).

La ligne (2) impose que les $b_{x,y}$ (à savoir "b avant y") soient des variables binaires.

La ligne (3) impose que pour toute paire d'éléments (x, y), soit on place x avant y soit on place y avant x dans le classement consensuel.

Enfin, la ligne (4) correspond à l'inégalité triangulaire : si on a placé x avant z et z avant z, alors x est placé avant y.

A nouveau, la prise en compte des égalités et des classements incomplets n'est pas gérée par cette formulation. Il est donc important de généraliser ces équations afin de les rendre compatibles dans un cadre plus général incluant

les classements avec égalités et les classements incomplets. Une tentative de généralisation a été proposée dans [25] pour gérer les classements complets avec égalités. Cependant, elle contient une erreur que nous corrigerons dans le Chapitre 4.

Algorithmes de type séparation et évaluation

Les algorithmes de type séparation et évaluation (*branch and bound* en anglais) sont fréquemment utilisés pour le problème d'agrégation de permutations [37, 38, 67, 69]. Ces algorithmes utilisent des connaissances sur les bornes supérieures du problème de minimisation afin d'éliminer des familles de solutions dont on sait qu'elles ne peuvent pas être optimales. Par exemple, les auteurs de [67] se servent des travaux de [35] sur le calcul de bornes pour le score de Kemeny minimal pour leur algorithme exact *branch and bound*.

Les auteurs de [37] ne se basent pas directement sur le score de Kemeny pour leur algorithme exact *branch and bound*, mais cherchent à maximiser la somme des coefficients de corrélations de Kendall [45] apès avoir montré que ce problème est équivalent à celui de minimiser le score de Kemeny.

Les algorithmes exacts ne peuvent être utilisés que sur de petites instances. Quel que soit le type d'algorithme exact considéré, leur utilisation n'est plus possible dès lors que le nombre d'éléments à trier dépasse quelques dizaines d'éléments pour des raisons de temps d'exécution [1].

2.1.3 Algorithmes d'approximation et heuristiques

Les algorithmes exacts étant difficilement utilisables sur des jeux de données contenant plusieurs centaines d'éléments comme c'est souvent le cas en biologie, il est alors nécessaire de se servir d'algorithmes d'approximation et d'heuristiques. Cette sous-section passe en revue les heuristiques principalement utilisées dans le cadre de l'agrégation de classements.

KwikSort : un algorithme diviser pour régner

L'algorithme KwikSort conçu par Ailon *et al.* [3] est une $2-$ approximation pour le problème d'agrégation de permutations : le score de Kemeny d'un classement consensuel renvoyé par cet algorithme est au pire deux fois plus élevé que le score de Kemeny d'un classement consensuel optimal. Cet algorithme est randomisé et son fonctionnement est inspiré du tri rapide.

Remarque. Dans le meilleur des cas, les pivots choisis ont systématiquement séparé en deux groupes de même taille les éléments restant à trier. La complexité est alors $\Theta(n * \log_2 n)$ où n est le nombre d'éléments de U. Dans le pire des cas, les pivots choisis ont systématiquement induit deux sous-ensembles dont l'un des deux est vide.

1. Il est possible d'aller jusqu'à une centaine d'éléments pour un algorithme de programmation linéaire en nombres entiers utilisant un solveur commercial comme Cplex ou Gurobi [19]

C'est le cas si, à chaque fois, le pivot choisi est le premier ou le dernier élément parmi ceux restant. La complexité est alors $\Theta(n^2)$.

Bien que cet algorithme soit une $2-$approximation de la méthode Kemeny-Young, son ratio d'approximation devient $\frac{11}{7}$ lorsqu'il est utilisé conjointement avec l'algorithme *Pick a Perm* également conçu par Ailon [3]. Ce dernier consiste à choisir comme classement consensuel la permutation de R qui minimise le score de Kemeny avec R.

Recherche locale

L'algorithme *BioConsert* présenté dans [32] est une heuristique de recherche locale. Cette heuristique a été comparée aux autres heuristiques et algorithmes d'approximation de l'état de l'art et a fourni les classements consensuels de meilleure qualité lors d'un banc d'essai effectué par Brancotte *et al.* [25] sur des jeux de données réels et synthétiques contenant des égalités.

Le principe de cette heuristique est le suivant : on part de chaque classement d'entrée et pour chaque élément on effectue un des deux mouvements possibles suivants si et seulement si ce mouvement améliore le score de Kemeny :

- déplacer l'élément dans un *bucket* existant
- déplacer l'élément dans un nouveau *bucket*

On effectue ces mouvements jusqu'à ce qu'on ne puisse plus effectuer de tels mouvements capables de diminuer le score de Kemeny. Le meilleur classement parcouru est ainsi retourné.

L'algorithme GRASP [43] est une autre heuristique gloutonne de type recherche locale, dont le fonctionnement est très proche de BioConsert. La différence principale est que GRASP est un algorithme randomisé contrairement à BioConsert qui est déterministe.

Borda, Copeland et Schulze : des systèmes électoraux à part entière

Les méthodes Borda [39] Copeland [36] et Schulze [79] sont trois systèmes électoraux qui prennent en entrée une liste de classements et renvoient un classement consensuel des candidats. Bien qu'étant des systèmes électoraux à part entière, les méthodes Borda et Copeland sont utilisées comme heuristiques pour l'agrégation de classements au sens de la minimisation du score de Kemeny (méthode Kemeny-Young). La méthode Schulze, introduite plus récemment, est beaucoup moins utilisée dans ce contexte mais présente l'avantage d'être axiomatiquement proche de la méthode Kemeny-Young [26] (l'ensemble des critères d'équité respectés par les deux méthodes est très proche).

Les méthodes Borda, Copeland et Schulze consistent à associer un score à chaque élément à classer. Le classement consensuel renvoyé correspond au tri par ordre décroissant de score des éléments.

Soient $R = [r_1, r_2, ..., r_m]$ une liste de permutations de U contenant n éléments et $x \in U$. On appelle $Bor(x)$ le

score associé à x par la méthode Borda. On a : $Bor(x) = \sum_{i=1}^{m} \ n - r_i[x]$ (pour rappel, $r_i[x]$ est la position de x dans r_i). En d'autres termes, $Bor(x)$ est la somme du nombre de candidats classés après x dans chaque classement d'entrée auquel on a retiré 1.

Illustration 2.1.1. *Considérons une liste* $R = [r_1, r_2, r_3, r_4, r_5]$ *constituée des cinq permutations suivantes :*

$r_1 = [\{A\}, \{B\}, \{C\}]$

$r_2 = [\{A\}, \{B\}, \{C\}]$

$r_3 = [\{A\}, \{B\}, \{C\}]$

$r_4 = [\{C\}, \{B\}, \{A\}]$

$r_5 = [\{C\}, \{B\}, \{A\}]$

On a $Bor(A) = 2+2+2+0+0 = 6$, $Bor(B) = 1+1+1+2+2 = 7$ *et* $Bor(C) = 0+0+0+2+2 = 4$. *Ainsi, le classement consensuel obtenu est* $r = [\{B\}, \{A\}, \{C\}]$.

Soient R une liste de permutations de U et $x \in U$. On appelle $Cop(x)$ le score associé à x par la méthode Copeland. On a : $Cop(x) = w(x) - l(x)$, avec $w(x)$ correspondant au nombre d'éléments y tels que x est avant y dans une stricte majorité de classements et $l(x)$ correspondant au nombre d'éléments y tels que y est avant x dans une stricte majorité de classements.

Si on considère la même liste de permutations que pour la méthode Borda, on obtient $Cop(A) = 2-0 = 2$ puisque A est avant B dans une stricte majorité de classements et avant C dans une stricte majorité de classements tandis qu'aucun élément n'est avant A dans une stricte majorité de classements. On a également $Cop(B) = 1 - 1 = 0$ puisque B est avant C dans une stricte majorité de classements mais après A dans une stricte majorité de classements. De la même manière, on a $Cop(C) = 0 - 2 = -2$. Ainsi, le classement consensuel obtenu est $r = [\{A\}, \{B\}, \{C\}]$. Ce classement consensuel est différent de celui obtenu par la méthode Borda.

La méthode Schulze utilise la construction de graphes ainsi que des algorithmes de calcul du plus court chemin afin d'associer son score à chaque élément. Les détails de cette méthode sont présentés dans [79].

Calculer les scores associés à chaque élément de U est de complexité $\Theta(n*m)$ par la méthode Borda, $\Theta(n^2*m)$ pour la méthode Copeland et $\Theta(n^3*m)$ pour la méthode Schulze (n étant la taille de U à savoir le nombre d'éléments à classer et m la taille de R à savoir le nombre de classements).

2.1.4 Récapitulatif des différents algorithmes utilisés pour l'agrégation de classements

Pour conclure cette section, nous présentons un tableau rassemblant les différents algorithmes utilisés dans le cadre de l'agrégation de classements.

1. Concevoir des algorithmes exacts en PLNE et *Branch and Bound* pour gérer les classements avec égalités et les classements incomplets nécessite une formulation très différente des algorithmes exacts pour les permutations.

Algorithme	Classements avec égalités (voir [25])	Généralisable aux données manquantes
ExactPLNE [35]	non [1]	non [1]
ExactBNB [38]	non [1]	non[1]
KwikSort [3]	oui	oui
MEDRank [46]	oui	non
Pick-A-Perm [3]	oui	non
RepeatChoice [2]	oui	non
BioConsert [32]	oui	oui
Borda [39]	oui	non
CopelandMethod [36]	oui	oui
SchulzeMethod [79]	oui	oui

TABLE 2.3 – Algorithmes couramment utilisés dans le cadre de l'agrégation de classements

Inconvénients des algorithmes d'approximation et heuristiques

Les heuristiques, par essence, n'offrent aucune garantie quant à la qualité du classement consensuel retourné dans le sens où ces méthodes ne permettent pas de savoir si son score associé est proche ou au contraire très éloigné du score d'un classement consensuel optimal. Les algorithmes d'approximation quant à eux ont un ratio d'approximation très élevé [3, 2] et peuvent donc aboutir à des classements consensuels dont le score de Kemeny bien supérieur au score d'un classement consensuel optimal, ou ne s'adaptent pas aux classements avec égalités [60]. Le banc d'essai présenté dans [25] montre que ces algorithmes d'approximation fournissent des classements consensuels de qualité inférieure à BioConsert à la fois sur les jeux de données réels et synthétiques utilisés. Par ailleurs, le ratio d'approximation des algorithmes d'approximation n'a été démontré que pour les permutations. Ces algorithmes sont donc de simples heuristiques lorsqu'ils sont adaptés pour gérer les classements avec égalités.

Sur des classements incomplets avec égalité contenant plusieurs centaines d'éléments, il est très difficile d'évaluer la qualité du classement consensuel. En effet, les algorithmes exacts ne permettent pas de calculer un classement consensuel en un temps raisonnable et de plus il n'y a pas d'algorithme d'approximation dans ce cadre. Nous allons voir dans la section suivante que des techniques de partitionnement étroitement liées à la théorie du choix social offrent une solution intermédiaire entre algorithmes exacts et heuristiques.

2.2 A l'interface entre algorithmique et choix social : les techniques de partitionnement

Nous avons vu dans la section précédente que les algorithmes exacts sont trop lents s'il y a beaucoup d'éléments à classer et que les heuristiques ne permettent pas d'évaluer la qualité du classement consensuel renvoyé. Une solution classiquement utilisée pour pallier ces problèmes est de diviser le problème initial en sous-problèmes indépendants de sorte que que la simple concaténation d'un classement consensuel optimal de chaque sous-problème forme un classement consensuel optimal pour le problème de départ.

Plusieurs techniques de partitionnement présentes dans la littérature et détaillées dans cette section permettent un tel partitionnement.

2.2.1 Truchon et le critère de Condorcet étendu

Truchon définit le *Critère de Condorcet étendu* dans [83] permettant de partitionner le problème de départ en sous-problèmes indépendants pour un nombre impair de classements. Comme son nom l'indique, ce critère est une extension du critère de Condorcet défini dans la Sous-section 2.2.3. Notons que Truchon permet les égalités dans les classements d'entrée et considère que le coût de rompre une égalité est égal à 1.

Définition 2.2.1. *Critère de Condorcet étendu et partition de Truchon [83]*

Truchon définit le critère de Condorcet étendu de la manière suivante. Soit R une liste contenant un nombre impair de classements complets de U et $\mathscr{T} = [T_1, ..., T_k]$ une partition ordonnée de U telle que $\forall i < j$ et $\forall (x, y) \in T_i \times T_j$, x est avant y dans une majorité de classements. Alors, $\forall i < j$ et $\forall (x, y) \in T_i \times T_j$, x est avant y dans tout classement consensuel optimal[2]. Une telle partition sera appelée partition de Truchon.

Reprenons nos trois permutations présentées en Table 2.1. On voit que A est avant tous les autres éléments dans une stricte majorité de classements. Ainsi, $\mathscr{T} = [\{A\}, \{B, C, D\}]$ est une partition de Truchon. On peut en déduire que A est avant B, C et D dans toutes les solutions optimales.

Il est important de préciser que même si les classements de départ peuvent contenir des égalités, l'objectif de Truchon est de rompre ces égalités afin de produire une permutation de U. Il est donc intéressant de se demander si le critère de Condorcet étendu peut être utilisé dans un cadre plus large, comme par exemple la distance de Kendall-τ généralisée avec pénalité p (voir Définition 1.3.1 et [47]).

Les auteurs de [19] déterminent une partition de Truchon en calculant les composantes fortement connexes d'un graphe. La complexité de cet algorithme est $\Theta(|R| * |U|^2)$.

2.2.2 Les éléments propres

Betlzer *et al.* définissent dans [17] le concept d'*éléments propres (non-dirty elements)* en proportion q.

Définition 2.2.2. *Un élément x est dit propre en proportion q si pour tout élément $y \neq x$, soit x est avant y dans une proportion de classements supérieure ou égale à q, soit y est avant x dans une proportion de classements supérieure ou égale à q.*

Cas q = 0.75. Les auteurs montrent que si x est un élément propre en proportion 0.75, alors tous les éléments placés avant x dans au moins 75% des permutations sont placés avant x dans tous les classements consensuels

2. Truchon considère dans [83] une adaptation du score de Kemeny équivalente au score de Kemeny avec pénalité p de Fagin *et al.* [47] en prenant $p = 1$.

optimaux et tous les éléments placés après x dans au moins 75% des permutations sont placés après x dans tous les classements consensuels optimaux. Ceci permet donc de calculer une partition ordonnée de U en trois parties $[B_1, B_2, B_3]$ avec B_1 correspondant à l'ensemble des éléments placés avant x dans au moins 75% des permutations, $B_2 = \{x\}$ et B_3 correspondant à l'ensemble des éléments placés après x dans au moins 75% des permutations. Intuitivement, si un élément y est avant x dans au moins 75% des permutations et qu'un élément z est après x dans au moins 75% des permutations, alors y est nécessairement avant z dans au moins la moitié des permutations.

Cas q = 2/3. Betlzer *et al.* ont également démontré dans [17] que si x est un élément propre en proportion $\frac{2}{3}$, alors il existe un classement consensuel optimal r^* tel que tous les éléments placés avant x dans au moins $\frac{2}{3}$ des permutations sont placés avant x dans r^* et tous les éléments placés après x dans au moins $\frac{2}{3}$ des permutations sont placés après x dans r^*.

Si l'objectif est de calculer un classement consensuel le plus rapidement possible, ce critère sera préféré au précédent. En effet, un élément propre en proportion 0.75 est nécessairement un élément propre en proportion $\frac{2}{3}$ mais la réciproque n'est pas nécessairement vraie. Les éléments propres en proportion $\frac{2}{3}$ sont donc susceptibles de partitionner en un plus grand nombre de parties. Cependant, il est tout à fait possible que des classements consensuels optimaux ne respectent pas le partitionnement obtenu avec les éléments propres en proportion $\frac{2}{3}$. Cette méthode de partitionnement n'est donc pas à utiliser si l'objectif est de mettre en lumière des points communs entre l'ensemble des classements consensuels optimaux.

Il est intéressant de noter que d'une part le critère de Condorcet étendu et d'autre part le critère de Betzler pour le cas $q = 0.75$ ont le même objectif à savoir calculer une partition ordonnée des éléments à trier telle que tout classement consensuel optimal respecte cette partition. Nous montrerons dans le Chapitre 3 que ces deux critères sont complémentaires. En effet, certains jeux de données peuvent être partitionnés par le critère de Condorcet étendu mais pas par les éléments propres tandis que d'autres peuvent au contraire être partitionnés par les éléments propres mais pas par le critère de Condorcet étendu.

Ces techniques de partitionnement sont particulièrement intéressantes à coupler aux algorithmes exacts afin de réduire leur temps d'exécution lorsque le nombre d'éléments à trier est grand. Si chaque sous-problème obtenu est de taille suffisamment petite pour qu'un algorithme exact soit utilisé, un classement consensuel optimal est alors obtenu. Malheureusement, ces techniques de partitionnement souffrent de deux défauts majeurs. Nous verrons dans le Chapitre 3 que le critère de Condorcet étendu et les éléments propres en proportion 75% permettent rarement un partitionnement en de nombreux sous-problèmes sur les jeux de données réels. Par ailleurs, aucune méthode de partitionnement décrite dans cette section n'est compatible avec la distance de Kendall-τ généralisée pour les classements avec égalités. A notre connaissance, aucune n'a été adaptée pour les classements incomplets avec égalités.

2.2.3 Lien entre techniques de partitionnement et critères d'équité en théorie du choix social

Nous avons vu dans la Sous-section 2.2.1 le critère de Condorcet étendu. Comme son nom l'indique, ce critère est une extension du critère de Condorcet qui est très couramment cité en théorie du choix social (au point que les systèmes électoraux sont dit *de Condorcet* s'ils respectent ce critère). Plus généralement, l'étude axiomatique qui intéresse la communauté théorie du choix social permet de caractériser le comportement des différentes méthodes d'agrégation de classements. Si la méthode considérée pour cette thèse est appelée *méthode Kemeny-Young* en théorie du choix social, il est important de préciser que d'autres méthodes d'agrégation de classements existent et correspondent à différents systèmes électoraux (méthode Copeland, règle de Borda, ...). Les critères d'équité permettent alors de caractériser mathématiquement ces différentes méthodes d'agrégation de classements. Nous allons nous focaliser sur deux critères d'équité qui ont un intérêt tout particulier dans le cadre de la recherche de classement consensuel optimal au sens du score de Kemeny : le *LIIA* pour *Local Independance of Irrelevant Alternatives* [85] et le *critère de Condorcet*. Plus précisément, nous allons voir que le *LIIA* fait du critère de Condorcet une technique de partitionnement du problème de départ en sous-problèmes indépendants.

Partitionnement et "Indépendance locale des alternatives non pertinentes"

Le critère d'équité appelé *Local Independence of Irrelevant Alternatives* abrégé *LIIA* et traduit en français par *Indépendance locale des alternatives non pertinentes* est un critère d'équité qui est respecté par la méthode Kemeny-Young dans le cas où les classements dont on cherche un classement consensuel sont des permutations de U [86].

Propriété 2.2.1. *Soit un sous-ensemble $S \subseteq U$ de k éléments tel que les éléments de S correspondent aux k premiers ou k derniers éléments dans un classement consensuel optimal r^*. Soient r' le classement r^* duquel seuls les éléments de S ont été conservés et R' la liste de classements de S correspondant à la liste de classements R dont seuls les éléments de S ont été conservés. Autrement dit, r' est la projection de r sur S et R' est la projection de R sur S. Alors, r' est un classement consensuel optimal pour R'.*

Illustration 2.2.1. *Pour illustrer le LIIA, prenons une liste $R = [r_1, r_2, r_3, r_4]$ constituée des quatre permutations suivantes :*

$r_1 = [\{A\}, \{C\}, \{D\}, \{B\}]$

$r_2 = [\{B\}, \{A\}, \{C\}, \{D\}]$

$r_3 = [\{A\}, \{B\}, \{D\}, \{C\}]$

$r_4 = [\{B\}, \{D\}, \{C\}, \{A\}]$

Un algorithme exact nous indique que $r^* = [\{A\}, \{B\}, \{C\}, \{D\}]$ est un classement consensuel optimal. Retirons maintenant l'élément D de ce dernier. On obtient $r' = [\{A\}, \{B\}, \{C\}]$. Grâce au LIIA, on sait que r' est un

classement consensuel optimal pour la liste $R' = [r'_1, r'_2, r'_3, r'_4]$ constituée des quatre permutations suivantes (les mêmes que précédemment, mais les D ont été retirés) :

$r'_1 = [\{A\}, \{C\}, \{B\}]$

$r'_2 = [\{B\}, \{A\}, \{C\}]$

$r'_3 = [\{A\}, \{B\}, \{C\}]$

$r'_4 = [\{B\}, \{C\}, \{A\}]$

Une conséquence directe du LIIA est la suivante. Prenons $S \subseteq U$ de taille k et supposons qu'une condition suffisante permette d'établir qu'il existe un classement consensuel optimal r^* pour R tels que l'ensemble des k premiers éléments de r^* soit égal à S (on n'a pas besoin de connaître l'ordre des éléments de S dans r^*). Appelons R_1 la liste de classements correspondant à R dont seuls les éléments de S ont été conservés et R_2 la liste de classements correspondant à R dont tous les éléments de S ont été retirés. Alors, concaténer un classement consensuel optimal de R_1 et un classement consensuel optimal de R_2 fournit un classement consensuel optimal pour R. Ainsi, grâce au LIIA, toute propriété garantissant qu'il existe un classement consensuel optimal commençant par un sous-ensemble de U donné permet de partitionner R en deux sous-problèmes indépendants.

Le fait que la méthode Kemeny-Young respecte le LIIA garantit également qu'il n'existe aucun algorithme polynomial permettant de trouver l'élément en première position d'un classement consensuel optimal sur n'importe quelle instance (à moins que $P = NP$). En effet, si un tel algorithme existait, il suffirait de trouver ce premier élément, de le retirer de R, puis de recommencer récursivement à chercher le premier élément pour le "nouveau R" et ainsi de suite. Ainsi, le classement consensuel optimal se construirait pas à pas, et on obtiendrait donc un classement consensuel optimal par un algorithme polynomial.

Partitionnement et "Critère de Condorcet"

Le critère de Condorcet est un critère fondamental pour caractériser un système de vote[3]. Nous en donnons la définition ci-dessous.

Définition 2.2.3 (Critère de Condorcet [34], vainqueur de Condorcet)**.** *Soit R une liste de permutations de U (ensemble de candidats à une élection). S'il existe un candidat $x \in U$ tel que pour tout autre candidat $y \neq x \in U$ on a x avant y dans une stricte majorité de classements, alors x est le vainqueur de l'élection. Un tel candidat est appelé vainqueur de Condorcet.*

Si ce critère a été défini dans un contexte de théorie du choix social, le fait qu'une méthode respecte (ou non) le critère reste évidemment valide quel que soit le contexte. La méthode Kemeny-Young respecte le critère de Condorcet : s'il y a un vainqueur de Condorcet, il est nécessairement en première position dans tous les classements consensuels optimaux [26]. Le critère de Condorcet établit donc une condition suffisante pour trouver le

3. Un système électoral respectant le critère de Condorcet est appelé *extension de Condorcet*.

vainqueur d'une élection ou plus généralement l'élément qui arrive en première position de tout classement consensuel optimal.

Ce critère combiné au LIIA offre donc une méthode de partitionnement : trouver un vainqueur de Condorcet permet de partitionner R en deux sous-problèmes indépendants R_1 et R_2 : R_1 correspond à R auquel on n'a gardé que l'élément x dans les classements et R_2 correspond à R auquel on a retiré x de tous les classements. Le LIIA garantit que le fait de retirer x des classements de R ne changera pas l'ordre relatif des autres éléments dans le(s) classement(s) consensuel(s) optimal/optimaux.

Illustration 2.2.2. *Pour illustrer le critère de Condorcet, prenons une liste $R = [r_1, r_2, r_3, r_4]$ constituée des quatre permutations suivantes :*

$r_1 = [\{C\}, \{A\}, \{D\}, \{B\}]$

$r_2 = [\{A\}, \{B\}, \{C\}, \{D\}]$

$r_3 = [\{A\}, \{B\}, \{D\}, \{C\}]$

$r_4 = [\{B\}, \{D\}, \{A\}, \{C\}]$

On peut voit que A est avant B dans une stricte majorité de permutations (r_1, r_2, r_3), A est avant C dans une stricte majorité de permutations (r_2, r_3, r_4) et A est avant D dans une stricte majorité de permutations (r_1, r_2, r_3). L'élément A est donc un vainqueur de Condorcet. On a donc la garantie que tous les classements consensuels optimaux commencent par A et que concaténer $[\{A\}]$ avec un classement consensuel optimal au sous-problème $R' = [r'_1, r'_2, r'_3, r'_4]$ ci-dessous donne un classement consensuel optimal pour R :

$r'_1 = [\{C\}, \{D\}, \{B\}]$

$r'_2 = [\{B\}, \{C\}, \{D\}]$

$r'_3 = [\{B\}, \{D\}, \{C\}]$

$r'_4 = [\{B\}, \{D\}, \{C\}]$

Notons que dans l'Illustration 2.2.2, il suffirait d'intervertir A et D dans r_1 pour que A ne soit plus un vainqueur de Condorcet : A ne serait plus avant D dans une stricte majorité de permutations (il n'y a donc aucun vainqueur de Condorcet dans ce cas là).

Nous avons vu de quelle manière les critères d'équité de la théorie du choix social peuvent être utilisés pour définir des techniques de partitionnement permettant de diviser le problème initial en sous-problèmes indépendants. Il est important de préciser que si la méthode Kemeny-Young respecte le LIIA et le critère de Condorcet qui sont à l'origine des techniques de partitionnement, le respect de ces critères n'est pas prouvé dans un contexte plus large que celui des permutations.

2.3 Gestion des données manquantes

Nous avons vu dans la section précédente qu'il existe de nombreux algorithmes et plusieurs techniques de réduction d'espace pour le problème d'agrégation de permutations. Or, lorsque les classements dont on souhaite déterminer un classement consensuel sont incomplets ou contiennent des égalités, il n'est plus possible d'utiliser la distance de Kendall-τ et le score de Kemeny tels qu'ils ont été définis initialement (voir Définitions 1.2.1 et 1.2.2). En conséquence, de nombreux travaux de recherche se focalisent sur les classements complets et sans égalités [2, 3, 5, 12, 13, 18, 19, 21, 38, 60, 62]. Pour autant, de nombreux jeux de données sont incomplets et contiennent des égalités. Nous avons vu dans le Chapitre 1 que le modèle de Fagin *et al.* permet de gérer les égalités. Ce modèle est très couramment utilisé dans la littérature. Pour la gestion des données manquantes, la situation est différente. Les auteurs de [6] ont adapté le coefficient de corrélation de [45] pour les classements incomplets. Pour l'agrégation de classements au sens de Kemeny, on trouve deux familles de méthodes permettant de gérer les données manquantes : d'une part celles consistant à rendre artificiellement complets des classements qui ne le sont pas à l'origine et d'autre part celles consistant à adapter le score de Kemeny. Il se trouve que de nombreuses adaptations du score de Kemeny ont été proposées [4, 17, 24, 44]. Nous allons d'abord étudier les méthodes consistant à rendre complets des classements qui ne le sont pas, puis nous nous intéresserons aux adaptations du score de Kemeny.

Afin d'illustrer les différentes méthodes, les deux classements ci-dessous seront réutilisés tout au cours de cette section.

$$r = [\{C\}, \{B\}]$$
$$s = [\{B\}, \{A\}, \{C\}, \{D\}]$$

TABLE 2.4 – Exemples de deux classements r et s de $U = \{A, B, C, D\}$. Le classement r est incomplet puisque A et D sont manquants.

2.3.1 Projection et unification

Nous nous intéressons ici à une première catégorie de méthodes permettant de gérer les classements incomplets. Les méthodes présentées ici consistent à rendre artificiellement complets des classements à l'origine incomplets. Ces méthodes sont au nombre de deux : *projection* et *unification*.

Projection

La projection est une des deux méthodes utilisées dans la littérature pour rendre complets des classements qui ne le sont pas [19]. Intuitivement, elle consiste à retirer des classements d'entrée tous les éléments qui ne sont pas

présents dans chaque classement d'entrée. Une définition plus formelle est donnée ci-dessous.

Définition 2.3.1. *La projection est une méthode consistant à prendre en entrée une liste* $R = [r_1, r_2, ..., r_m]$ *de* m *classements possiblement incomplets de* U *et de renvoyer une liste* $R' = [r'_1, r'_2, ...r'_m]$ *de* m *classements complets de* U' *telle que :*

- $U' = \bigcap_{i \leq m} dom(r_i)$.
- $\forall i \leq m, \forall x \neq y \in U', x \prec_{r_i} y \Rightarrow x \prec_{r'_i} y$.
- $\forall i \leq m, \forall x \neq y \in U', x \equiv_{r_i} y \Rightarrow x \equiv_{r'_i} y$.

Illustration 2.3.1. *Considérons la liste de classements* $R = [r_1, r_2, r_3]$ *avec*

$r_1 = [\{C, D\}, \{B\}]$,

$r_2 = [\{B\}, \{C\}, \{E\}]$,

$r_3 = [\{A\}, \{B, C\}, \{D\}]$.

On peut observer que les classement r_1 *et* r_2 *sont incomplets puisque* A *y est absent alors qu'il est présent dans* r_3*. De la même manière,* r_2 *est incomplet puisque* A *et* D *sont manquants. On observe que seuls les éléments* B *et* C *sont présents dans chaque classement. Appliquer la technique de projection revient à calculer* $R' = [r'_1, r'_2, r'_3]$ *avec*

$r'_1 = [\{C\}, \{B\}]$,

$r'_2 = [\{B\}, \{C\}]$,

$r'_3 = [\{B, C\}]$.

Cette méthode a été utilisée par Betzler *et. al* sur des jeux de données réels, en particulier des résultats de courses de F1 et des résultats de requêtes sur des moteurs de recherche [19].

L'inconvénient majeur de cette méthode est la perte d'information qui en découle. En effet, un coureur absent d'une seule course de F1 n'apparaîtra pas dans le classement consensuel même s'il a fini premier de toutes les autres courses.

Unification

L'unification consiste à rendre complets les classements d'entrée en ajoutant à la fin de chaque classement incomplet un *bucket d'unification* contenant l'ensemble des éléments absents du classement.

Une définition plus formelle est donnée ci-dessous.

Définition 2.3.2. *L'unification est une méthode consistant à prendre en entrée une liste* $R = [r_1, r_2, ..., r_m]$ *de classements possiblement incomplets de* U *et de renvoyer une liste* $R' = [r'_1, r'_2, ...r'_m]$ *de classements complets de* U *telle que pour tout* $i \leq m$ *on a* $r'_i = r_i$ *si* $dom(r_i) = U$ *et* $r'_i = r_i \cup (U \backslash dom(r_i))$ *sinon.*

Cette technique a été utilisée dans [78] et [32].

Illustration 2.3.2. *Considérons la liste de classements* $R = [r_1, r_2, r_3]$ *avec*

$r_1 = [\{A\}, \{C, D\}]$,

$r_2 = [\{B\}, \{C\}]$,

$r_3 = [\{A\}, \{B, C\}, \{D\}]$.

On peut observer que le classement r_1 *est incomplet puisque* B *y est absent (alors qu'il est présent dans* r_2 *et* r_3*). De la même manière,* r_2 *est incomplet puisque* A *et* D *sont manquants. Le classement* r_3 *est en revanche complet. Appliquer la technique d'unification sur* R *revient à considérer la nouvelle liste de classements* $R_u = [r'_1, r'_2, r'_3]$ *avec*

$r'_1 = [\{A\}, \{C, D\}, \{B\}_u]$,

$r'_2 = [\{B\}, \{C\}, \{A, D\}_u]$,

$r'_3 = [\{A\}, \{B, C\}, \{D\}] = r_3$ *(pas de changement puisque* r_3 *est déjà complet).*

Pour insister sur le fait que r'_1 et r'_2 sont issus de classements initialement incomplets, leurs *buckets* d'unification portent un u en indice.

Utilisation des techniques de projection et d'unification sur des jeux de données Une fois les classements rendus complets par projection ou unification, les techniques d'agrégation de classements complets peuvent s'appliquer. L'inconvénient majeur de cette méthode est qu'elle nécessite deux hypothèses fortes sur les données pour pouvoir l'appliquer. D'une part il faut considérer que les éléments manquants sont moins pertinents que les éléments présents vis-à-vis d'un classement et d'autre part il faut considérer que l'ensemble des éléments absents sont d'une non-pertinence équivalente puisque tous les éléments manquants se retrouvent à égalité dans un même groupe d'unification.

Sur l'exemple de classements incomplets présentés en Table 2.4, on obtient après unification que le score de Kemeny généralisé (avec pénalité p) $S^p(r, s) = 1_{(A,C)} + p_{(A,D)} + 1_{(B,C)} = 2 + p$

2.3.2 Généralisations de la distance de Kendall-τ et du score de Kemeny

Nous nous focalisons maintenant sur une deuxième catégorie de méthodes permettant de gérer les classements incomplets. Les méthodes présentées ici consistent à généraliser la distance de Kendall-τ définissant ainsi une nouvelle mesure de dissimilarité entre deux classements [4]. Certaines de ces généralisations prennent également en compte les égalités et d'autres non.

4. Les mesures définies ne sont généralement pas des distances au sens mathématique du terme

Mesure de Kendall-τ induite

Dwork *et al.* définissent la *mesure de Kendall-τ induite* qui calcule une mesure de dissimilarité entre deux classements. Cette mesure (qui n'est pas une fonction de distance puisqu'elle ne respecte ni l'identité ni l'inégalité triangulaire) s'applique aux classements incomplets sans égalités. Le principe est de calculer la distance de Kendall-τ entre les deux classements pris en argument mais en ne considérant que les paires d'éléments présents dans les deux classements. La définition formelle est donnée ci-dessous.

Définition 2.3.3 (Mesure de Kendall-τ induite [44])**.** *La mesure de Kendall-τ induite entre deux classements incomplets sans égalités r_1 et r_2 notée $J(r_1, r_2)$ est définie de la façon suivante :*

$$J(r_1, r_2) = |\{(x, y) \in (dom(r_1) \cap dom(r_2))^2 : x \prec_{r_1} y \wedge y \prec_{r_2} x\}| \tag{2.1}$$

Par exemple, pour les classements de la Table 2.4, on a $J(r, s) = 1_{(B,C)} = 1$. Pour cette mesure, on ne considère en effet que les éléments communs à r et s. La mesure de Kendall-τ induite est donc équivalente à la distance de Kendall-τ pour les deux classements suivants :

$r' = [\{C\}, \{B\}]$

$s' = [\{B\}, \{C\}]$

Définition 2.3.4 (Score de Kemeny induit)**.** *Soit U l'ensemble des éléments à trier. Le score de Kemeny induit entre une permutation r de U et une liste R de classements sans égalités de U (complets ou incomplets) noté $S^I(r, R)$ est défini de la façon suivante :*

$$S^I(r, R) = \sum_{r_i \in R} J(r, r_i) \tag{2.2}$$

Cette définition a été reprise par Betzler *et al.* dans [17] sous le nom de *Kemeny score with incomplete votes*.

Notons que si pour la mesure de Kendall-τ induite comme pour la technique de projection il est question de considérer seulement les éléments en commun, les deux techniques restent très différentes l'une de l'autre. En effet, lors du calcul du score de Kemeny induit entre r et R, une même paire d'éléments peut être ignorée pour certains classements d'entrée mais être considérée pour d'autres classements d'entrée. Les éléments absents de certains classements sont ici considérés et apparaissent dans le classement consensuel final contrairement à ce qui est obtenu par la technique de projection.

Pseudo-distance de Kendall-τ généralisée

Une autre généralisation de la distance de Kendall-τ appelée *pseudo-distance de Kendall-τ généralisée*[5] a été définie par Brancotte *et al.* dans [24]. Cette pseudo-distance consiste à utiliser la technique d'unification mais

5. Une pseudo-distance respecte le critère de symétrie ainsi que l'inégalité triangulaire mais pas l'identité.

sans considérer que les éléments du *bucket* d'unification sont nécessairement ex-aequo. Notons que cette pseudo-distance reprend également le modèle proposé par Fagin *et al.* [47] pour traiter les égalités.

Définition 2.3.5 (Pseudo-distance de Kendall-τ généralisée [24])**.** *Soient U l'ensemble des éléments à trier, r_1 et r_2 deux classements de U, $u(r_1)$ le bucket d'unification de r_1 (possiblement vide), $u(r_2)$ le bucket d'unification de r_2 (possiblement vide) et $p \in [0,1]$. La pseudo-distance de Kendall-τ généralisée, notée $N^p(r_1, r_2)$, est définie de la manière suivante :*

$$\begin{aligned} N^p(r_1, r_2) =& |\{(x,y) \in U^2 : x \prec_{r_1 \bigcup u(r_1)} y \wedge y \prec_{r_2 \bigcup u(r_2)} x| \\ &+ p * |(x,y) \in U^2 : x \equiv_{r_1} y \wedge (x \notin u(r_1) \vee y \notin u(r_1)) \wedge x \not\equiv_{r_2} y| \\ &+ p * |(x,y) \in U^2 : x \equiv_{r_2} y \wedge (x \notin u(r_2) \vee y \notin u(r_2)) \wedge x \not\equiv_{r_1} y\}| \end{aligned} \tag{2.3}$$

Illustration 2.3.3. *Reprenons les classements r et s de la Table 2.4. On a $N^p(r,s) = 1_{(B,C)} + 1_{(A,C)} = 2$. Pour cette mesure, on considère que les éléments absents de r sont "virtuellement" à la fin de r_1 mais sans supposer d'ordre entre eux. Ainsi, il faut une pénalité de 1 entre A et C dans la mesure où C est (virtuellement) avant A dans r alors que A est avant C dans s. Par contre, il n'y a pas de pénalité due à la paire (A, D) puisque ni A ni D ne sont présents dans r.*

L'inconvénient de cette formulation décrite dans [24] est qu'elle nécessite que les classements soient passés par le processus d'unification afin de les rendre complets vis-à-vis de U. Les auteurs présentent cette pseudo-distance comme un moyen de réduire le biais induit par le processus d'unification qui positionne artificiellement des éléments à égalité. En effet, en considérant cette pseudo-distance, une paire d'éléments à égalité dans un seul des deux classements induira un coût nul si la relation d'égalité est au sein du *bucket* d'unification.

A partir de cette pseudo-distance, Brancotte *et al.* ont également généralisé le score de Kemeny entre une liste de classements R et un classement consensuel r.

Définition 2.3.6 (Score de Kemeny associé à la pseudo-distance de Kendall-τ généralisée)**.** *Le score de Kemeny associé à la pseudo-distance de Kendall-τ généralisée entre une liste de classements R et un classement consensuel r noté $M^p(r, R)$ est défini de la manière suivante :*

$$M^p(r, R) = \sum_{r_i \in R} N^p(r, r_i) \tag{2.4}$$

L'heuristique BioConsert est adaptée [24] pour être rendue compatible avec la pseudo-distance de Kendall-tau généralisée.

Distance de Kendall-τ étendue

Plus récemment, la *distance de Kendall-τ étendue* a été définie par Aledo *et al.* [4]. Cette mesure (qui n'est pas une fonction de distance puisqu'elle ne respecte ni l'identité, ni l'inégalité triangulaire) permet de gérer les classements incomplets avec égalités. Cette mesure est une extension de la mesure de Kendall-τ induite [44] (voir Définition 2.3.3).

Définition 2.3.7 (Distance de Kendall-τ étendue [4])**.** *La distance de Kendall-τ étendue entre deux classements r_1 et r_2, notée $B(r_1, r_2)$, est définie de la manière suivante :*

$$B(r_1, r_2) = |\{(x, y) \in (dom(r_1) \cap dom(r_2))^2 : x \prec_{r_1} y \wedge y \prec_{r_2} x\}| \tag{2.5}$$

Si les auteurs n'ont pas adapté explicitement le score de Kemeny à la distance de Kendall-τ étendue, ils définissent le problème d'agrégation de classements incomplets comme la recherche du (des) classement(s) minimisant la somme des distance de Kendall-τ étendue avec chaque classement en entrée. Pour plus de clarté dans les chapitres suivants, nous définissons le score de Kemeny étendu ci-dessous.

Définition 2.3.8 (Score de Kemeny étendu)**.** *Le score de Kemeny étendu entre un classement complet r de U et une liste de classements R, noté $H(r, R)$ est défini de la manière suivante :*

$$H(r, R) = \sum_{r_i \in R} B(r, r_i) \tag{2.6}$$

Si l'équation 2.5 s'écrit de la même manière que l'équation 2.1, elles n'ont pas été définie dans le même contexte. En effet, la première a été pensée pour gérer des classements sans égalités alors que cette dernière gère les classements avec égalités et considère que le coût de créer / rompre une égalité est nul.

L'heuristique GRASP est adaptée dans [4] pour la rendre compatible avec la distance de Kendall-τ étendue.

Nous avons vu que plusieurs adaptations du score de Kemeny existent dans la littérature. Ces différentes adaptations donnent un résultat différent sur les mêmes classements pris en exemple. Erick Moreno-Centeno et Adolfo Escobedo proposent dans [71] une étude axiomatique permettant de savoir si une adaptation de la distance de Kendall-τ pour les classements incomplets respecte certaines propriétés comme la commutativité ou l'inégalité triangulaire. Malgré ces travaux, il est très difficile de savoir quelle adaptation il est préférable de choisir sur un jeu de données réel contenant des classements incomplets avec égalités. De plus, ces adaptations ne sont pas associées à des propriétés mathématiques permettant d'utiliser des techniques de réduction d'espace, et aucun algorithme exact n'a été proposé pour l'une d'entre-elles. Enfin, très peu d'heuristiques ont été conçues pour répondre au besoin d'agrégation de classements incomplets.

2.4 Conclusion : enjeux pour l'agrégation de classements

Faciliter l'utilisation d'algorithmes exacts est un véritable enjeu pour lesquels plusieurs travaux de recherche ont abouti à des résultats. En effet, des techniques de partitionnement permettent de diviser le problème initial en sous-problèmes indépendants. Ces techniques ont néanmoins deux défauts majeurs. Premièrement, elles ne sont définies que sur des classements complets [83], voire uniquement sur des permutations [19, 20]. Deuxièmement, les conditions suffisantes permettant le partitionnement sont strictes et permettent peu de partitionner en pratique. Un premier enjeu est donc de trouver des conditions suffisantes plus souples et adaptés aux classements incomplets permettant de mieux partitionner efficacement les jeux de données avec données manquantes. L'objectif est ici de fournir rapidement un classement consensuel de qualité.

Par ailleurs, les heuristiques de l'état de l'art ne fournissent aucune information quant à la qualité du classement consensuel renvoyé. Les algorithmes d'approximation ont un ratio d'approximation trop élevé pour que le classement consensuel renvoyé soit nécessairement de bonne qualité. Le deuxième enjeu est donc de développer des heuristiques capables de fournir des indicateurs quant à la fiabilité du classement consensuel calculé.

Une troisième problématique peu étudiée dans la littérature est celle concernant la non-unicité du classement consensuel optimal. Il est pourtant très important pour un utilisateur de savoir à quel point le classement consensuel renvoyé est robuste, en particulier sur les premiers éléments du classement consensuel. Le troisième enjeu est donc de trouver des propriétés mathématiques permettant de rendre compte des points communs entre les solutions optimales.

Enfin, la gestion des classements incomplets pose problème pour deux raisons. Premièrement, plusieurs méthodes existent pour étendre la méthode Kemeny-Young aux classements incomplets sans qu'un utilisateur puisse être informé des conséquences que peut avoir le choix d'une méthode plutôt qu'une autre sur le classement consensuel calculé ou sur le temps d'exécution des algorithmes. Deuxièmement, aucune propriété mathématique n'est associée à ces différentes méthodes, ne rendant possible que l'utilisation d'heuristiques dont la fiabilité est inconnue. Le quatrième et dernier enjeu est de définir un modèle associé à des propriétés mathématiques et englobant les différentes méthodes d'agrégation de classements incomplets. Le but est ici double : fournir à l'utilisateur un guide lui permettant de choisir la méthode la plus adaptée à ses besoins et proposer des algorithmes permettant de renvoyer un classement consensuel de qualité.

Dans le Chapitre 3, nous généralisons une technique de réduction d'espace pour la pseudo-distance de Kendall-τ généralisée et nous nous attaquons au problème de la multiplicité des classements consensuels optimaux. Le Chapitre 4 présente un modèle pour les classements incomplets incluant les différentes généralisations du score de Kemeny présentées dans ce chapitre ainsi qu'un cadre algorithmique et axiomatique associés à ce modèle. Ces contributions sont évaluées sur un grand nombre de jeux de données réels et synthétiques.

Résumé du Chapitre 2

Dans ce chapitre nous avons rappelé le lien très étroit entre le problème d'agrégation de permutations et le *weighted feedback arcset problem*, problème très classique en théorie des graphes [44, 59, 60] qui consiste à rendre un graphe acyclique en retirant des arêtes dont la somme des poids doit être minimale.

Tous les travaux concernant le *weighted feedback arcset problem* sont directement utilisables pour l'agrégation de permutations. Ce lien entre les deux problèmes n'est plus valable sur les classements avec égalités ou les classements incomplets.

Pour résoudre le problème d'agrégation qui est NP-difficile, plusieurs algorithmes exacts ont été proposés [5, 11, 30, 35, 38, 69, 78, 84]. Les plus courants sont basés sur de la programmation linéaire en nombre entiers ou sur une approche de type séparation et évaluation. En pratique, ces approches ne peuvent pas être utilisées lorsque le nombre d'éléments à trier est supérieur à quelques dizaines. En conséquence, plusieurs algorithmes d'approximation et heuristiques ont été conçus [2, 3, 32, 36, 39, 46, 79]. Ces méthodes souffrent de deux inconvénients majeurs : elles n'offrent pas ou que très peu de garanties sur la qualité du classement consensuel renvoyé (les ratios d'approximation des algorithmes d'approximation sont élevés) et la grande majorité ne considèrent que les classements complets. Trois heuristiques ont un statut particulier : Borda, CopelandMethod et SchulzeMethod qui correspondent à des systèmes électoraux et n'ont pas été conçues pour calculer un classement consensuel au sens de la minimisation du score de Kemeny. Les deux premières sont néanmoins très couramment utilisées dans ce contexte [25, 44, 56].

Les critères d'équité de la théorie du choix social offrent alors des outils pour aider au calcul de classements consensuels de bonne qualité dans le cadre des permutations. Celui appelé LIIA (indépendance locale des alternatives non pertinentes) garantit que si on retire les k premiers éléments d'un classement consensuel optimal, ce qu'il reste de ce classement est un classement consensuel optimal pour les classements de départ dont on a retiré ces mêmes éléments. Ce critère est étroitement lié au critère de Condorcet étendu [83] et aux éléments propres [17] : le fait de les associer permet de partitionner le problème initial en sous-problèmes indépendants. Ces deux méthodes de partitionnement ne sont définies que dans le cadre de l'agrégation de classements complets.

Nous avons également fait un état de l'art des méthodes existantes pour gérer les classements incomplets. On les divise en deux familles : celles qui consistent à rendre artificiellement complets les classements qui ne le sont pas et celles qui généralisent le score de Kemeny. Dans la première famille, on trouve la projection qui consiste à retirer des classements tous les éléments qui ne sont pas présents dans chaque classement [19] et l'unification qui consiste à ajouter à la fin de chaque classement incomplet un *bucket d'unification* contenant tous les éléments manquants [32, 78]. Dans la deuxième famille, plusieurs généralisations ont été proposées [4, 17, 24, 44] mais la pertinence de leur utilisation sur des données réelles n'a pas été démontrée. Aucun algorithme exact ni aucune technique de réduction d'espace n'ont été publiées pour ces généralisations du score de Kemeny.

Chapitre 3

Partitionnement à base de graphes et critère de robustesse

Sommaire

3.1 Introduction

Dans ce chapitre, nous nous plaçons dans un contexte biologique précis dans lequel les classements sont possiblement incomplets et avec égalités. Nous considérons pour ce chapitre le score de Kemeny généralisé à la pseudo-distance [24] (voir Définition 2.3.6) qui a été définie pour ce contexte [24]. Nous commençons par décrire le contexte biologique considéré pour ce chapitre, puis nous résumons les contributions algorithmiques apportées.

3.1.1 Contexte biologique

Les contributions de ce chapitre seront illustrées à travers le cas d'utilisation présenté en introduction correspondant à la recherche de gènes impliqués dans une maladie donnée. Lorsqu'on interroge les bases de données biologiques de référence pour trouver les gènes les plus impliqués dans une maladie donnée, on obtient des classements de gènes très différents selon la dénomination choisie pour la maladie d'intérêt (le nombre de gènes renvoyés est différent, un même gène peut être positionné très différemment d'une reformulation à l'autre, ...). Partons d'une maladie et considérons ses différents synonymes (par exemple : *breast cancer*, *breast carcinoma*, *mammary cancer*, ...). Pour chaque synonyme, le classement des gènes les plus pertinents est obtenu en interrogeant la base de données *Gene* du NCBI. Il est alors intéressant d'utiliser l'agrégation de classements puisque cela permet d'obtenir un classement à partir de l'information apportée par l'ensemble des reformualations synonymes. Ainsi, si un

gène impliqué dans une maladie est très peu (ou pas) associé à une reformulation, il peut quand même être bien positionné dans le classement consensuel.

Si les méthodes présentées dans ce chapitre sont utilisables dans différents contextes, elles ont un intérêt particulier dans ce contexte biologique qui souffre de trois problèmes majeurs exposés au chapitre précédent.

- Problème 1 : le nombre de gènes à classer est généralement très grand et il n'y a pas de technique de réduction d'espace pour les classements incomplets. Il est donc impossible de se servir d'algorithmes exacts.
- Problème 2 : dans ce contexte, l'adaptation du score de Kemeny utilisée pour gérer les classements incomplets avec égalités est le score de Kemeny adapté à la pseudo-distance [24] (Définition 2.3.6) . L'heuristique BioConsert est la seule qui a été adaptée pour cette pseudo-distance. Or, elle ne fournit aucune garantie quant à la qualité des classements consensuels renvoyés.
- Problème 3 : le nombre de classements consensuels optimaux peut être très élevé, posant alors le problème de la représentativité d'un classement consensuel unique vis-à-vis du jeu de données.

3.1.2 Contributions

Ce chapitre reprend nos contributions pour répondre aux problèmes listés ci-dessus. Ces contributions sont publiées dans [7] et [8]. Pour répondre aux deux premiers problèmes soulevés, nous avons conçu et développé ParCons, une méthode à base de graphes permettant de calculer rapidement des classements consensuels sur des jeux de données de très grande taille en les décomposant en des jeux de données beaucoup plus petits et indépendants où des algorithmes de qualité (éventuellement exacts) peuvent être utilisés. Par ailleurs, pour répondre au troisième problème, nous avons conçu et développé ParFront, un algorithme capable d'établir des frontières entre des groupes de gènes de telle sorte que leur ordre relatif dans l'ensemble des classements consensuels optimaux soit inviolable. Ces frontières donnent donc à l'utilisateur des informations sur la *robustesse* des positions des gènes dans le classement consensuel calculé.

Les algorithmes ParCons et ParFront ont été implémentés dans l'outil ConQuR-BioV2 afin de les rendre disponibles pour la communauté des Sciences de la Vie.

Dans ce qui suit, la Section 3.2 présente un exemple repris tout le long du chapitre pour illustrer les différents concepts et algorithmes décrits. La Section 3.3 décrit une manière de représenter les classements d'entrée en se focalisant sur les paires d'éléments, propice à l'utilisation de graphes. La Section 3.4 définit un nouveau graphe adapté au score de Kemeny généralisé à la pseudo-distance (Définition 2.3.6) utilisé dans ce contexte. La Section 3.5 présente ParCons répondant aux problèmes 1 et 2. La Section 3.6 présente ParFront répondant au problème 3. La Section 3.7 décrit ConQuR-BioV2, l'outil en ligne dans lequel l'ensemble des méthodes décrites dans ce chapitre sont implémentées. La Section 3.8 correspond à une évaluation quantitative des algorithmes ParCons et ParFront sur des jeux de données biologiques correspondant au contexte décrit dans la sous-section précédente. La Section

3.9 correspond à une évaluation qualitative des classements consensuels calculés par ParCons. Cette évaluation utilise des *gold standards* obtenus à partir de la base de données Orphanet qui est une encyclopédie de référence en ce qui concerne les maladies rares. Enfin, la Section 3.10 dresse les conclusions de ce chapitre.

3.2 Exemple et intuition quant aux enjeux

Les classements obtenus dans notre contexte de recherche de gènes impliqués dans une maladie donnée ont des caractéristiques particulières. Après avoir présenté une liste de classements nous servant d'exemple tout au long de ce chapitre, nous nous servons de cet exemple pour donner une intuition de la manière dont (i) les gènes peuvent être divisés en sous-groupes facilitant ainsi le calcul d'un classement consensuel, (ii) on peut évaluer la robustesse du classement consensuel obtenu.

3.2.1 Un exemple inspiré par les données biologiques

L'exemple présenté en Table 3.1 est inspiré des données biologiques décrites dans la Section 3.1.1 : U est ici l'ensemble $\{A, B, ..., I\}$ correspondant aux gènes d'intérêt pour une maladie donnée et les différents classements correspondent aux classements des gènes les plus liés à chaque reformulation synonyme de cette maladie lorsqu'on interroge la base de données *Gene* du NCBI. Afin de mimer au mieux les données biologiques, les classements présentés sont incomplets et avec égalités. Par exemple, les gènes D et E sont à égalité dans r_1, r_2, r_5, r_6. Le classement r_5 est incomplet car les gènes F, G et H y sont manquants. Les *buckets* d'unification ont été représentés en gris à la fin des classements incomplets.

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

TABLE 3.1 – Exemple simplifié de classements incomplets obtenus en interrogeant la base de données *Gene* du NCBI avec plusieurs synonymes d'une même maladie.

3.2.2 Une première intuition quant au calcul du classement consensuel

Actuellement, aucune méthode ne permet de partitionner un tel problème en sous-problèmes puisque les techniques de partitionnement décrites dans la Section 2.2 ne sont pas définies sur les classements incomplets.

Nous donnons dans cette sous-section des intuitions quant à la manière dont un classement consensuel de bonne qualité peut être calculé avec l'idée de partitionner le problème initial en sous-problèmes. On peut remarquer

en particulier trois points particulièrement intéressants.

Premièrement, D et E sont devant les gènes A, B, C, F, G, H, I dans une stricte majorité de classements. En conséquence, un classement consensuel raisonnable devrait alors placer D et E aux deux premières positions.

Deuxièmement, I est avant A, B, C, G, H dans une stricte majorité de classements et avant F dans une majorité de classements : un classement consensuel raisonnable devrait alors placer I avant $\{A, B, C, F, G, H\}$. Troisièmement, A, B et C sont toujours placés avant G et H. De plus, A, B et C sont placés avant F dans la majorité des classements. Encore une fois, un classement consensuel raisonnable devrait placer A, B, C avant F, G et H.

Cette toute première analyse permet de mettre en évidence la présence de quatre groupes de gènes distincts. Appelons groupe 1 celui de D, E, groupe 2 celui de I, groupe 3 celui de A, B, C et groupe 4 celui de F, G, H.

Inspectons maintenant ces quatre groupes plus en détail pour tenter de classer les gènes au sein des différents groupes.

Groupe 1 (D et E). Comme indiqué ci-dessus, D et E sont à égalité dans une stricte majorité de classements. On peut conclure qu'avoir D et E à égalité dans le classement consensuel semble être un choix judicieux.

Groupe 2 (I). Comme I est seul dans son groupe, classer ce gène au sein de ce groupe est trivial.

Groupe 3 (A, B et C). Analysons ces gènes par paires. Dans une stricte majorité de classements (4 contre 2), A est avant B. Il peut donc sembler naturel d'avoir A avant B dans un classement consensuel. Cependant, le problème suivant se produit : B est avant C dans une stricte majorité de classements (4 contre 2) et C est avant A dans une stricte majorité de classements (4 contre 2). Il est pourtant impossible de satisfaire les trois contraintes A avant B, B avant C et C avant A en même temps. Aucun classement consensuel trivial ne ressort de ce groupe.

Groupe 4 (F, G et H). Dans une stricte majorité de classements, F est avant G (4 contre 2). À nouveau, il peut sembler naturel d'attendre F avant G dans un classement consensuel. À l'inverse, il existe une "relation d'indifférence" pour la paire $\{F, H\}$: dans la moitié des classements F est avant H et dans la moitié des classements H est avant F. Par conséquent, dans ce cas, l'ordre relatif entre F et H sera d'une certaine manière arbitraire. La situation est similaire pour G et H. Enfin, placer F avant G semble être un choix fiable, et placer H avant F, entre F et G ou après G semblent être des choix équivalents.

Nous verrons dans la Section 3.5 que $[\{D, E\}, \{I\}, \{B\}, \{C\}, \{A\}, \{F\}, \{G\}, \{H\}]$ est un classement consensuel optimal selon le score de Kemeny généralisé à la pseudo-distance [24] (voir Définition 2.3.6) pour la liste de classements de cet exemple.

3.2.3 Intuition quant à l'évaluation de la robustesse des classements consensuels

Comme vu ci-dessus, la position de H semble avoir plus de flexibilité que la position de D et E. En fait $[\{D, E\}, \{I\}, \{B\}, \{C\}, \{A\}, \{H\}, \{F\}, \{G\}]$ est un autre classement consensuel optimal possible. Un algorithme exact naïf énumérant tous les classements consensuels possibles indique qu'il y a un total de 9 classements consensuels optimaux pour cet exemple. La mise en évidence des points communs entre classements consensuels optimaux est donc particulièrement importante pour fournir à l'utilisateur une information sur les gènes robustes, c'est-à-dire les gènes dont les positions ne varient pas beaucoup dans l'ensemble des classements consensuels optimaux.

Nous verrons dans la Section 3.5 que le fait d'avoir D et E avant tous les gènes restants dans une stricte majorité de classements nous permet de dire que D et E sont positionnés avant A, B, C, F, G, H, I dans tous les classements consensuels optimaux. Afin de formaliser ce concept, nous avons défini le concept de *frontière* dans la Section 3.6.

L'intuition donnée dans cette section est qu'en regardant les gènes paire par paire, il est possible de diviser ces gènes en sous-groupes formant ainsi des sous-problèmes. La section suivante formalise cette intuition.

3.3 Représentation des classements par paires d'éléments

L'exemple présenté a abouti au partitionnement des éléments en quatre groupes. Ce partitionnement effectué à la main est basé sur une analyse paire par paire des éléments. Malheureusement, la formulation du score de Kemeny généralisé à la pseudo-distance présentée dans [24] (voir Définition 2.3.6) ne permet pas de rendre compte immédiatement de l'impact d'une paire d'éléments donnée sur le score d'un classement consensuel. Après avoir introduit des définitions préliminaires dans la Sous-section 3.3.1, nous proposerons dans la Sous-section 3.3.2 une formulation alternative pour le score de Kemeny généralisé à la pseudo-distance permettant de mesurer l'impact de chaque paire d'éléments sur le score final. Enfin, nous utilisons dans la Sous-section 3.3.3 cette formulation alternative et présentons une matrice indiquant pour chaque paire (x, y) d'éléments le coût des différents ordres relatifs possibles dans le classement consensuel. Cette matrice est particulièrement utile aux algorithmes ParCons et ParFront présentés dans les sections suivantes de ce chapitre.

3.3.1 Définitions préliminaires

Dans l'introduction de ce chapitre, nous avons observé certains points communs entre les classements d'entrée pour déterminer intuitivement de quelle manière le problème initial pouvait être partitionné. En fait, la question à se poser est la suivante : étant donné une paire d'éléments (x, y), quel est le coût de placer x avant y, y avant x ou x à égalité avec y dans le classement consensuel vis-à-vis du score de Kemeny généralisé à la pseudo-distance ?

Nous définissons ci-dessous les fonctions $before$ et $tied$ permettant de calculer ces coûts.

Définition 3.3.1 ($before^p(x, y)$)**.** *Soient x et y deux éléments de U, R une liste de classements et $p \in [0, 1]$. La fonction $before^p(x, y)$ mesure le coût de mettre x avant y dans un classement consensuel selon la pseudo-distance de Kendall-τ généralisée. Elle a pour valeur :*

$$\begin{aligned} before^p(x,y) =& |\{r \in R : y \prec_r x \vee (x \notin dom(r) \wedge y \in dom(r)\}| \\ &+ p * |\{r \in R : x \equiv_r y\}| \end{aligned} \tag{3.1}$$

D'après la définition de $before^p(x, y)$, pour calculer le coût de mettre x avant y dans le classement consensuel, on ajoute une pénalité de

- 1 pour chaque classement positionnant y avant x ou dans lequel y est présent et x absent
- p pour chaque classement positionnant x et y à égalité

Illustration 3.3.1. *Prenons la paire (D, E) de notre exemple. Dans quatre classements, D et E sont à égalité et dans un classement, E est avant D. Il n'y a aucun classement tel que E est présent et D absent. Ainsi, $before^p(D, E) = 1 + 4 * p$ correspond au coût de positionner D devant E dans le classement consensuel.*

De la même manière, on peut voir que $before^p(E, D) = 1 + 4 * p$. Le coût de placer le gène D avant le gène E dans le classement consensuel est donc le même que celui de placer E avant D.

Définition 3.3.2 ($tied^p(x, y)$)**.** *Soient x et y deux éléments de U, R une liste de classements et $p \in [0, 1]$. La fonction $tied^p(x, y)$ mesure le coût de mettre x et y à égalité dans un classement consensuel selon la pseudo-distance de Kendall-τ généralisée. Elle a pour valeur :*

$$tied^p(x,y) = p * |\{r \in R : (x \in dom(r) \vee y \in dom(r)) \wedge x \not\equiv_r y\}| \tag{3.2}$$

D'après la définition de $tied^p(x, y)$, pour calculer le coût de mettre x et y à égalité dans le classement consensuel, on ajoute une pénalité de p pour chaque classement ne positionnant pas x et y à égalité (en excluant les classements pour lesquels x et y sont tous les deux absents).

Illustration 3.3.2. *Reprenons la paire (D, E) de notre exemple. Dans quatre classements D et E sont à égalité, dans un classement E est avant D et dans un classement D est avant E. Ainsi, $tied^p(D, E) = p * 2$ correspond au coût de positionner D et E à égalité dans le classement consensuel.*

Enfin, nous définissons $min^p(x, y)$ comme étant le coût du positionnement relatif le moins coûteux pour la paire (x, y).

Définition 3.3.3 ($min^p(x, y)$)**.** *Soient x et y deux éléments de U, R une liste de classements et $p \in [0, 1]$. La fonction $min^p(x, y)$ mesure le coût du positionnement relatif le moins coûteux pour la paire (x, y). Elle a pour valeur :*

$$min^p(x,y) = min(before^p(x,y), before^p(y,x), tied^p(x,y))) \tag{3.3}$$

Illustration 3.3.3. *Pour la paire (D,E), on a $before(D,E) = 1+4*p$, $before(E,D) = 1+4*p$ et $tied(D,E) = 2*p$. Si on prend par exemple $p=1$ ou $p=0.5$, on obtient qu'il est plus intéressant de mettre D et E à égalité plutôt que D avant E ou E avant D.*

3.3.2 Nouvelle formulation du score de Kemeny associé à la pseudo-distance

Ces fonctions préliminaires permettent une nouvelle formulation pour le score de Kemeny généralisé à la pseudo-distance. Cette formulation permet de calculer le score en effectuant une somme sur les paires d'éléments à trier. Ainsi, l'impact de chaque paire sur le calcul du score est rendu plus visible.

Notation. *Nous notons $\mathscr{P}(U)$ l'ensemble des paires non ordonnées de U.*

Définition 3.3.4 (Score de Kemeny associé à la pseudo-distance de Kendall-τ généralisée, nouvelle formulation)**.** *Le score de Kemeny associé à la pseudo-distance de Kendall-τ généralisée entre un classement c qui est un classement complet de U (classement consensuel) et une liste R de classements de U, noté $M^p(c,R)$, est défini de la manière suivante.*

$$M^p(c,R) = \sum_{(x,y)\in\mathscr{P}(U)} \overline{M_c^p}(x,y) \tag{3.4}$$

avec $\overline{M_c^p}(x,y) =$

- $before^p(x,y)$ *si* x *est avant* y *dans* c
- $before^p(y,x)$ *si* y *est avant* x *dans* c
- $tied^p(x,y)$ *si* x *et* y *sont à égalité dans* c

Pour la suite de ce chapitre et afin d'illustrer les différentes étapes qui vont suivre sur des exemples concrets, nous posons $p=1$ [*]. Ainsi, nous écrirons $M(c,R)$ pour $M^1(c,R)$, $before(x,y)$ pour $before^1(x,y)$, $tied(x,y)$ pour $tied^1(x,y)$ et $min(x,y)$ pour $min^1(x,y)$.

Tous les résultats qui suivent restent valables pour n'importe quelle valeur de p.

3.3.3 Matrice des coûts par paires

Après avoir défini $before(x,y)$ (respectivement $tied(x,y)$) comme étant le coût de placer x avant y (respectivement x à égalité avec y) dans le classement consensuel, nous obtenons la *matrice de coût des éléments par paires*

[*]. Ce choix, bien que récurrent dans la littérature, est arbitraire et sera discuté au chapitre suivant.

en calculant pour chaque paire d'éléments (x, y) les valeurs respectives de $before(x, y)$, $before(y, x)$, $tied(x, y)$ et $min(x, y)$.

Illustration de la matrice des coûts des éléments par paires. La Table 3.2 représente la matrice des coûts des éléments par paires sur l'exemple présenté Table 3.1. Dans un souci de lisibilité, toutes les paires ne sont pas représentées. Pour chaque paire représentée, les coûts minimaux sont représentés en gras.

TABLE 3.2 – Fragment de la matrice de coût des éléments par paires pour l'exemple de la Table 3.1.

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	**1**	4	5	1
I	F	**3**	**3**	6	3
G	H	**2**	**2**	4	2

On peut remarquer un lien très fort entre cette matrice et les intuitions présentées dans la Sous-section 3.2.2 sur le calcul du classement consensuel. Considérons un élément x parmi ceux du groupe 1 (D, E) et un gène y parmi ceux d'un autre groupe. On peut observer si on complète la Table 3.2 que le coût de mettre x avant y est plus bas que le coût de mettre y avant x ou x à égalité avec y (autrement dit, $before(x, y) = min(x, y)$). On observe également que $tied(D, E) = min(D, E)$ alors que $before(D, E) > min(D, E)$ et $before(E, D) > min(D, E)$. Nous pouvons en conclure qu'il est strictement moins coûteux de mettre à égalité D et E dans un classement consensuel plutôt que d'avoir D avant E ou E avant D.

Ceci explique intuitivement que l'on retrouve D et E à égalité et en tête dans les classements consensuels optimaux.

Cette matrice nous permet de connaître le ou les ordres relatifs préférentiels pour chaque paire d'éléments. Cette vision par paires nous permet de représenter la liste de classements dont on cherche un classement consensuel par un graphe orienté dont la présence d'un arc d'un élément x vers un autre élément y dépend de la ou des positions relatives les moins coûteuses pour la paire (x, y).

3.4 Un nouveau graphe pour représenter les classements

Cette section a pour objectif de définir un graphe dont les composantes fortement connexes correspondent au partitionnement en sous-problèmes indépendants du problème de départ. Nous démontrerons dans la section suivante que la concaténation d'un classement consensuel optimal de chaque sous-problème suivant un ordre facile à déterminer nous fournit un classement consensuel optimal pour le problème de départ, répondant ainsi aux

problèmes 1 et 2 présentés en introduction de ce chapitre.

Dans la Sous-section 3.4.1, nous définissions G_e, un graphe défini à partir de la matrice des coûts par paires permettant ainsi de représenter les classements d'entrée de façon pertinente vis-à-vis du score de Kemeny généralisé à la pseudo-distance et donc du contexte biologique présenté en introduction de ce chapitre. La Sous-section 3.4.2 définit G^c le graphe des composantes de G_e dont la propriété d'être acyclique le rend propice au partitionnement.

3.4.1 G_e : graphe compatible avec le score de Kemeny généralisé à la pseudo-distance

Définition du graphe des éléments G_e

Nous définissons G_e le *graphe des éléments* dont l'objectif est de fournir une représentation condensée de la matrice de coût par paires présentée en Table 3.2. Ses sommets sont les éléments à classer et pour chaque paire d'éléments (x, y), les arcs dépendent des coûts minimaux parmi $before(x, y)$, $before(y, x)$ et $tied(x, y)$. Plus précisément, une absence d'arc de x vers y signifie intuitivement qu'il est raisonnable d'avoir y avant x dans un classement consensuel car cette absence d'arc signifie que $before(y, x)$ est minimal. Notons qu'il n'y a pas obligation que $before(y, x)$ soit l'unique coût minimal pour la paire (x, y).

La définition formelle de G_e est donnée ci-dessous.

Définition 3.4.1 (G_e, graphe des éléments)**.** *$G_e = (V_e, A_e)$ est le graphe orienté tel que*

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

Soit une paire d'éléments (x, y). La *Table des arcs de G_e* (Table 3.3) récapitule les différentes possibilités menant à un arc de x vers y ou de y vers x.

TABLE 3.3 – Table des arcs de G_e : arcs de G_e selon la positon de min(x,y) dans la matrice des coûts par paires.

$before(x, y)$	$before(y, x)$	$tied(x, y)$	dans G_e
= min(x,y)	$> min(x, y)$	$> min(x, y)$	$x \to y$
$> min(x, y)$	**= min(x,y)**	$> min(x, y)$	$x \leftarrow y$
$> min(x, y)$	$> min(x, y)$	**= min(x,y)**	$x \leftrightarrows y$
= min(x,y)	$> min(x, y)$	**= min(x,y)**	$x \to y$
$> min(x, y)$	**= min(x,y)**	**= min(x,y)**	$x \leftarrow y$
= min(x,y)	**= min(x,y)**	$> min(x, y)$	$x \quad y$
= min(x,y)	**= min(x,y)**	**= min(x,y)**	$x \quad y$

Illustration. La Figure 3.1 correspond au graphe G_e pour notre exemple.

Les cycles dans G_e : témoins de "conflits".

Les arcs orientés de G_e donnent une idée de quels éléments on voudrait voir avant les autres dans le classement consensuel. En effet, quand il y a un arc de x à y mais pas d'arc de y à x, alors $before(x, y) = min(x, y) \neq$

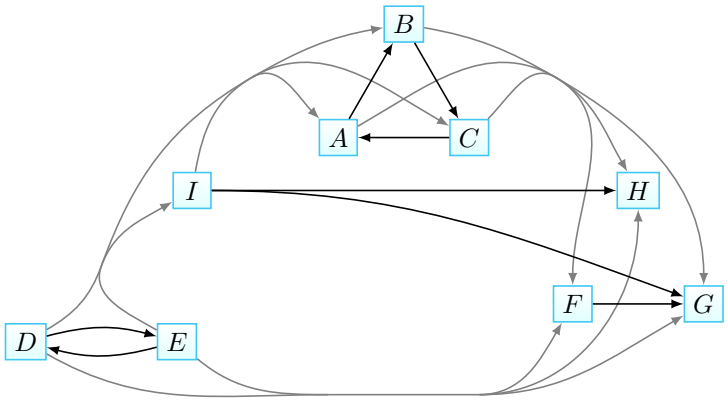


FIGURE 3.1 – Graphe des éléments G_e pour l'exemple de la Table 3.1. Les arcs gris avec sources/cibles multiples indiquent la présence d'un arc de chaque sommet de chaque source vers chaque sommet de chaque cible (par exemple, il y a un arc depuis chaque sommet de (D, E) vers chaque autre sommet, mais il n'y a pas d'arc de I à F).

$before(y, x)$ (voir Table des arcs de G_e, Table 3.3). Cependant, il est impossible de suivre chaque arc de G_e pour construire un classement consensuel : certains cycles de G_e rendent impossible la construction d'un classement car ils sont incompatibles avec la contrainte de transitivité. Pour reprendre le graphe G_e présenté en Figure 3.1, suivre l'arc de A vers B indique qu'on place A avant B, suivre l'arc de B vers C indique qu'on place B avant C et suivre l'arc de C vers A indique qu'on place C avant A. Il est donc impossible de suivre ces trois arcs.

Pour obtenir un graphe acyclique, nous allons calculer le graphe des composantes fortement connexes de G_e.

3.4.2 G^c : graphe des composantes fortement connexes de G_e

Cette sous-section rappelle quelques définitions de la théorie des graphes, puis explique pourquoi il est utile de calculer les composantes fortement connexes de G_e.

Rappelons qu'une composante fortement connexe (CFC) d'un graphe orienté $G = (V, A)$ est un sous-ensemble V' de V (éventuellement V lui-même) tel que (i) pour deux sommets quelconques (x, y) de V', il existe un chemin de x à y, et (ii) V' est maximal pour (i) c'est-à-dire qu'il n'y a pas de sous-ensemble V'' de V tel que $V' \subset V''$ et V'' respecte (i).

Définition 3.4.2 (G^c, graphe des composantes fortement connexes de G_e)**.** *On note $G^c = (V^c, A^c)$ le graphe des composantes fortement connexes de G_e c'est-à-dire le graphe orienté tel que :*

- *V^c est l'ensemble des composantes fortement connexes de G_e.*
- *$(c_i, c_j) \in A^c$ si et seulement s'il y a au moins un élément x de c_i et un élément y de c_j tel que (x, y) est un arc de G_e.*

Le calcul des composantes fortement connexes de G_e peut être effectué avec l'algorithme des composantes fortement connexes de Tarjan de complexité $O(|V_e| + |A_e|)$ [82].

Par définition, G^c est un graphe dirigé acyclique (DAG). Par conséquent, il existe au moins un tri topologique de G^c. Pour rappel, un tri topologique d'un DAG $G = (V, A)$ est une partition ordonnée $\mathscr{T} = [T_1, T_2, \ldots, T_k]$ de V telle que pour chaque $i < j$, $x \in T_i$, $y \in T_j$, on a $(y, x) \notin A$. Il peut y avoir plusieurs tris topologiques pour un même graphe dirigé acyclique. Notons qu'un tri topologique peut être calculé avec l'algorithme de Kahn de complexité $O(|V| + |A|)$ [57].

Illustration. La Figure 3.2 représente G^c le graphe des composantes fortement connexes de G_e pour notre exemple (Table 3.1). Le calcul des CFC de G_e permet la décomposition de l'ensemble des gènes à trier en 6 ensembles : $\{D, E\}$ (groupe 1), qui ne doivent pas être séparés car ils sont à égalité dans une stricte majorité de classements, $\{I\}$ (groupe 2), $\{A, B, C\}$ (groupe 3) qui forment une incompatibilité difficile à résoudre, et enfin $\{F\}$, $\{G\}$ et $\{H\}$ qui correspondent au groupe 4 que l'on a pu morceler en trois nouveaux sous-groupes.

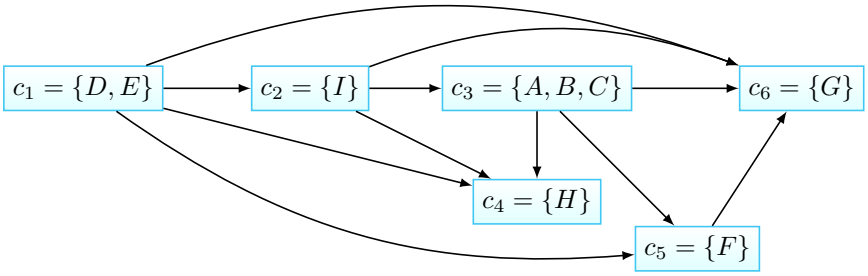


FIGURE 3.2 – G^c, le graphe des composantes fortement connexes de G_e.

Intérêt de G^c. Les sommets du graphe G^c forment une partition des gènes à trier. On peut le voir ainsi comme une décomposition du problème initial en sous-problèmes. Puisque G^c est acyclique, on peut en calculer au moins un tri topologique. Cela permet de classer ces ensembles de gènes avec un coût minimal vis-à-vis de la fonction de score à minimiser (puisque les arcs de G_e sont construits selon des coûts minimaux).

3.5 Utilisation du graphe G^c pour calculer un classement consensuel

L'objectif de cette section est d'utiliser le graphe G^c pour aboutir au calcul d'un classement consensuel. Nous commençons par démontrer dans la Sous-section 3.5.1 que la partition des éléments formée par les sommets de G^c définit des sous-problèmes indépendants répondant ainsi au problème 1. Dans la Sous-section 3.5.2, nous présentons ParCons, notre algorithme capable de renvoyer un classement consensuel de qualité en un temps raisonnable répondant ainsi au problème 2.

3.5.1 Propriété fondamentale de G^c

La propriété de G^c présentée dans cette section peut être exploitée pour partitionner le problème initial en plusieurs sous-problèmes permettant de calculer un classement consensuel de bonne qualité en un temps raisonnable. En effet, si les sous-problèmes obtenus sont de taille suffisamment petite, il devient possible d'utiliser un algorithme exact pour obtenir un classement consensuel optimal pour le sous-problème considéré.

Commençons par définir deux points de vocabulaire.

Définition 3.5.1 (Respect d'une partition ordonnée par un classement consensuel)**.** *On dira qu'un classement consensuel c respecte une partition ordonnée $T = [T_1, T_2, ..., T_k]$ de U si et seulement si pour toutes les paires d'éléments (x, y) telles que $x \in T_i$, $y \in T_j$ et $i < j$ alors x est avant y dans c.*

Illustration 3.5.1. *Considérons le tri topologique suivant pour le graphe G^c :*

$T = [\{D, E\}, \{I\}, \{A, B, C\}, \{F\}, \{H\}, \{G\}]$.

Le classement consensuel $c = [\{D, E\}, \{I\}, \{A\}, \{B\}, \{C\}, \{F\}, \{H\}, \{G\}]$ respecte T.

A l'inverse, le classement consensuel $c = [\{D, E\}, \{A\}, \{I\}, \{B\}, \{C\}, \{F\}, \{H\}, \{G\}]$ ne respecte pas T puisque A est avant I dans c alors que le sous-ensemble contenant A est après le sous-ensemble contenant I dans T.

Notation. On notera $R(T_i)$ la projection de R sur T_i c'est-à-dire la liste de classements R dont tous les éléments absents de T_i ont été retirés.

Illustration. Prenons comme point de départ la liste de classements R de notre exemple. Considérons c_1 la composante fortement connexe de G_e qui contient D et E. Alors, on obtient que $R(c_1) = R(\{D, E\}) = [r_1^{'}, r_2^{'}, r_3^{'}, r_4^{'}, r_5^{'}, r_6^{'}]$ où $r_1^{'} := [\{D, E\}]$, $r_2^{'} := [\{D, E\}]$, $r_3^{'} := [\{E\}, \{D\}]$, $r_4^{'} := [\{D\}, \{E\}]$, $r_5^{'} := [\{D, E\}]$, $r_6^{'} := [\{D, E\}]$.

Propriété 3.5.1 (Propriété de concaténation)**.** *Soit $\mathscr{T} = [T_1, T_2, \ldots, T_k]$ un tri topologique de G^c et μ_i un classement consensuel optimal pour $R(T_i)$. La concaténation $\mu_1.\mu_2 \ldots \mu_k$ est un classement consensuel optimal pour R.*

Démonstration :

Soit μ la concaténation $\mu_1.\mu_2 \ldots \mu_k$ et c un classement consensuel quelconque. Nous prouvons que μ est un classement consensuel optimal en montrant que $M(\mu, R) \leq M(c, R)$. Pour rappel, M correspond au score de Kemeny associé à la pseudo-distance de Kendall-τ généralisée (voir Équation 3.4).

Le score M est présenté dans l'équation 3.4 comme une somme sur $(x, y) \in \mathscr{P}(U)$. Dans un premier temps, pour chaque entier i de 1 à k, considérons les paires (x, y) d'éléments telles que x et y appartiennent tous les deux à T_i. Ces paires induisent un coût inférieur pour μ que pour c puisque μ_i est un classement consensuel optimal pour $R(T_i)$. Regardons maintenant les paires (x, y) d'éléments telles que x et y appartiennent à des parties différentes de $\mathscr{T}$. Supposons que $x \in T_i$, $y \in T_j$ avec $i < j$ (la preuve est similaire si $i > j$). Comme T_i est avant T_j dans $\mathscr{T}$, il n'y a pas d'arc de y à x dans G_e. Par construction de G_e, on peut conclure que

$before(x, y) = min(x, y)$. En d'autres termes, le coût induit par (x, y) dans μ ne peut pas être supérieur au coût induit par (x, y) dans c. On peut donc conclure que $M(\mu, R) \leq M(c, R)$ c'est-à-dire que μ est un classement consensuel optimal.

Corollaire 3.5.1. *Pour tout tri topologique $\mathscr{T} = [T_1, ..., T_k]$ de G^c, il existe un classement classement consensuel optimal qui respecte $\mathscr{T}$.*

Remarque. *La propriété de concaténation prouve l'existence d'un classement consensuel optimal respectant un tri topologique donné. Pour autant, il peut exister un classement consensuel optimal qui ne respecte aucun tri topologique. Un exemple est donné ci-dessous.*

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

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

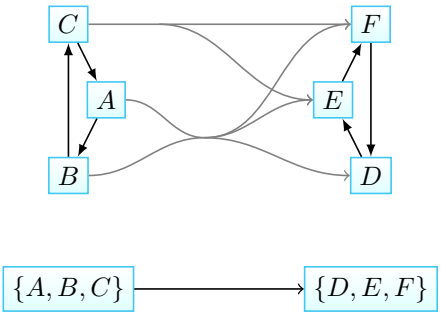


TABLE 3.4 – A gauche : exemple de liste de classements (en haut) et un classement consensuel optimal (en bas). A droite, le graphe G_e (en haut) et son graphe des composantes G^c (en bas). Le classement consensuel optimal ne respecte pas l'unique tri topologique de G^c puisque D est avant C dans le classement consensuel optimal.

Inspectons les classements présentés dans la Table 3.4.

Il y a deux composantes fortement connexes à savoir $\{A, B, C\}$ et $\{D, E, F\}$. En effet, aucun élément parmi $\{D, E, F\}$ ne peut atteindre un élément de $\{A, B, C\}$.

Par ailleurs, $[\{A\}, \{B\}, \{C\}]$ est un classement consensuel optimal pour le sous-problème induit par $\{A, B, C\}$ et $[\{D\}, \{E\}, \{F\}]$ est un classement consensuel optimal pour le sous-problème induit par $\{D, E, F\}$. Comme C qui est le dernier élément de $[\{A\}, \{B\}, \{C\}]$ n'a pas d'arc vers D qui est lui le premier élément de $[\{D\}, \{E\}, \{F\}]$, alors C et D peuvent être intervertis dans le classement consensuel optimal $[\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}]$. En définitive, $[\{A\}, \{B\}, \{D\}, \{C\}, \{E\}, \{F\}]$ est également un classement consensuel optimal bien qu'il ne respecte aucun tri topologique de G^c.

Une conséquence directe du Corollaire 3.5.1 est que le nombre de tris topologiques donne un minorant du nombre de classements consensuels optimaux.

Corollaire 3.5.2. *Le nombre de tri topologiques du graphe G^c est inférieur ou égal au nombre de classements consensuels optimaux.*

Le lien avec le problème 3 est très fort. Plus le nombre de tris topologiques est élevé, plus le nombre de classements consensuels optimaux l'est aussi. Il devient alors difficile pour un classement consensuel, même optimal,

d'être fortement représentatif de l'ensemble des classements d'entrée.

Une conséquence directe de la propriété de concaténation est que les différents sous-problèmes $R(T_1)$, $R(T_2)$, ..., $R(T_k)$ peuvent être traités indépendamment. Cette observation conduit naturellement à l'algorithme ParCons que nous présentons maintenant.

3.5.2 Un algorithme de partitionnement pour l'agrégation de classements

Grâce à la *propriété de concaténation*, nous savons qu'avoir un classement consensuel optimal pour chaque sous-problème obtenu grâce à G^c garantit l'obtention d'un classement consensuel optimal pour le problème initial. Pour répondre au problème 2, il faut maintenant être capable de calculer un classement consensuel pour chaque sous-problème ainsi obtenu. La question est donc maintenant de savoir comment traiter les éléments dans chaque CFC de G_e. Deux cas peuvent être envisagés. Si (cas 1), pour toute paire d'éléments (x, y) dans la CFC on a $tied(x, y) = min(x, y)$, alors le classement plaçant tous les éléments de la CFC à égalité est un classement consensuel optimal pour le sous-problème considéré.

Sinon (cas 2), un algorithme auxiliaire de calcul de classement consensuel doit être utilisé afin d'obtenir un classement consensuel pour le sous-problème considéré. Si la CFC est de petite taille, on utilise un algorithme exact. Dans le cas contraire, on utilise une heuristique ou un algorithme d'approximation.

L'algorithme ParCons décrit la procédure aboutissant à un classement consensuel. Il est important de noter que le résultat du classement consensuel renvoyé peut dépendre des algorithmes auxiliaires utilisés pour résoudre les sous-problèmes. En particulier, si un algorithme exact a été appelé pour chaque sous-problème, alors le classement consensuel renvoyé par ParCons est nécessairement optimal.

Illustration. Retournons vers notre exemple afin de construire pas à pas un classement consensuel tel que le fait l'algorithme ParCons. Un tri topologique possible pour G^c est : $[c_1, c_2, c_3, c_5, c_6, c_4]$ c'est-à-dire $[\{D, E\}, \{I\}, \{A, B, C\}, \{F\}, \{G\}, \{H\}]$.

Compte tenu de cet ordre, il est désormais possible de construire rapidement plusieurs parties du classement final consensuel de R. En effet, on sait que : (i) $[\{D, E\}]$, (ii) $[\{I\}]$, (iii) $[\{F\}]$, (iv) $[\{G\}]$ et (v) $[\{H\}]$ sont des classements consensuels optimaux pour les sous-problèmes induits respectivement par (i) c_1, (ii) c_2, (iii) c_5, (iv) c_6 et (v) c_4 (cas 1).

Même si positionner les gènes A, B et C dans le classement consensuel les uns par rapport aux autres n'est pas trivial (cas 2), on peut déjà affirmer qu'il existe un classement consensuel optimal dans lequel (i) D et E sont à égalité en première position ; (ii) I est placé en position 3 ; (iii) A, B et C partagent les positions 4, 5 et 6 sans qu'on sache en dire plus pour le moment (iv) F est en position 7 ; (v) G est en position 8 et (vi) H est en position 9.

Supposons maintenant que l'algorithme utilisé pour résoudre le sous-problème induit par c_3 renvoie comme classement consensuel $[\{A\}, \{B\}, \{C\}]$. Alors, le classement consensuel obtenu pour notre exemple est le suivant :

Algorithme 1 : ParCons : Procédure à base de graphe calculant un classement consensuel à l'aide d'un pré-traitement et d'un algorithme auxiliaire.

Input : R : set of rankings
begin
$M \leftarrow$ PairwiseCostMatrix(R)
$G_e \leftarrow$ ComputeGraphGe(M)
/* Calcul du graphe des CFC de G_e */
$G^c \leftarrow$ CFC(G_e)
/* Calcul d'un tri topologique $T = [T_1, \ldots, T_k]$ de G^c */
$topolSortGc \leftarrow$ TopologicalSorting(G^c)
/* Nombre de parties dans T */
$nbCFCGe \leftarrow$ numberCFCofGe
$consensus \leftarrow$ EmptyListOfSets()
/* Pour chaque composante fortement connexe de G_e */
for $i \leftarrow 1$ **to** *nbCFCGe* **do**
/* Si on trouve une paire d'éléments dont le coût de les mettre à égalité n'est pas minimal, on crée le sous-problème induit par la CFC considérée et on fait appel à un algorithme auxiliaire pour calculer un classement consensuel du sous-problème */
if $\exists v1 \neq v2$ **in** *topolSortGc[i]* **with** $tied(v1, v2) > min(v1, v2)$ **then**
$subProblem \leftarrow$ CopyInput(R)
Retire de $subProblem$ tous les éléments absents de $topolSortGe[i]$
$subConsensus \leftarrow$ auxiliaryAlgo(subProblem)
Ajoute chaque bucket de $subConsensus$ à consensus
/* Si toutes les paires d'éléments de la CFC considérée peuvent être mises à égalité avec un coût minimal, on le fait et on n'a alors pas besoin d'algorithme auxiliaire */
else
Ajoute $topolSortGc[i]$ à $consensus$
Result : $consensus$ (classement consensuel)

$[\{D, E\}, \{I\}, \{A\}, \{B\}, \{C\}, \{G\}, \{F\}, \{H\}]$.

L'algorithme ParCons permet donc de partitionner le problème initial en sous-problèmes indépendants y compris lorsque les classements d'entrée sont incomplets ou contiennent des égalités. Cet algorithme répond donc au problème 1 évoqué en introduction de ce chapitre. La propriété de concaténation associé à ParCons offre des garanties plus solides qu'une simple heuristique quant au score associé au classement consensuel renvoyé répondant ainsi au problème 2. Pour autant, le problème 3 reste entier. En effet, un des corollaires de cette section précise qu'il y a au moins autant de classements consensuels optimaux qu'il n'y a de tris topologiques de G^c. Sur des jeux de données réels, ce nombre peut être très élevé. Dans un tel cas, il est donc essentiel de savoir si certaines "zones" du classement consensuel renvoyé sont particulièrement fiables dans le sens où elles sont particulièrement représentatives des classements d'entrée.

Complexité de l'algorithme ParCons

La construction de la matrice des coûts par paires (Sous-section 3.3.3) est de complexité $O(|R| * |U|^2)$. En effet, pour chaque paire (x, y) d'éléments, on doit connaître leur position relative dans les classements de R afin de déterminer les valeurs de $before(x, y)$, $before(y, x)$ et $tied(x, y)$. La construction du graphe G_e se fait conjointement à la construction de la matrice, et étant donnée une paire (x, y) ainsi que les valeurs de $before(x, y)$, $before(y, x)$ et $tied(x, y)$, savoir s'il y a un arc de x vers y et/ou de y à x est de complexité $O(1)$.

Le calcul des composantes fortement connexes de G_e (algorithme de Tarjan [82]) est de complexité $O(|V_e| + |A_e|)$, c'est à dire $O(|U|^2)$ au pire.

Le calcul du tri topologique de G^c (algorithme de Kahn [57]) est également de complexité $O(|V_e| + |A_e|)$, c'est à dire $O(|U|^2)$ au pire.

L'algorithme de partitionnement du problème initial en sous-problèmes indépendant est donc de $O(|R| * |U|^2)$.

3.6 Frontières et partitionnement robuste

Alors que la Section 3.5 était consacrée au problème du calcul efficace d'un classement consensuel, cette section se concentre sur le problème de la non unicité du classement consensuel optimal. L'objectif est de rendre compte de points communs entre tous les classements consensuels optimaux afin qu'un utilisateur puisse juger de la pertinence du classement consensuel qui lui est renvoyé. Alors que ce problème a un très fort intérêt pratique, très peu de travaux de recherche s'y sont attaqués, et, de ce que nous savons, les classements incomplets n'ont encore jamais été considérés sur cette problématique. Les auteurs de [73] suggèrent de calculer l'ensemble des classements consensuels optimaux puis de calculer un classement consensuel des classements consensuels optimaux en utilisant une méthode de type Borda qui renvoie nécessairement un classement consensuel unique. L'inconvénient de cette méthode est que sur de nombreuses instances, il est impossible de calculer ne serait-ce

qu'un classement consensuel optimal. Notre objectif est donc de déterminer des points communs entre tous les classements consensuels optimaux sans les calculer. Pour ce faire, nous introduisons la notion de *frontière* afin d'identifier les zones robustes dans le classement consensuel retourné. Plus précisément, il est question de trouver des entiers $f_1, f_2, \ldots$ tels que l'ensemble des éléments dont la position est inférieure à f_i est identique dans tous les classements consensuels optimaux quelque soit i. Les frontières permettent donc de mettre en évidence pour l'utilisateur des groupes d'éléments dont la présence avant d'autres groupes d'éléments peut être considérée comme particulièrement justifiée. Une conséquence directe est la capacité à trouver des éléments dont la position varie très peu voire pas du tout dans tous les classements consensuels optimaux. Après avoir défini formellement la notion de frontière, nous présentons une première méthode de calcul de frontières basée sur un nouveau graphe dont la définition est proche de G_e. Dans un deuxième temps, nous présentons l'algorithme ParFront qui calcule des frontières en combinant l'information apportée par le graphe G_e et le nouveau graphe. Enfin, nous montrons que toutes les frontières calculées à partir de la première méthode sont également trouvées par ParFront.

3.6.1 Notion de frontière

L'objectif de cette section est de rendre compte de points communs entre tous les classements consensuels optimaux. Pour ce faire, nous introduisons la notion de *frontière* dont nous donnons une définition formelle ci-dessous.

Définition 3.6.1 (Frontière)**.** *Soient R une liste de classements, $\mathscr{C}^* = [c_1^*, c_2^*, \ldots]$ la liste des classements consensuels optimaux pour R et k un entier. k est une frontière pour R si et seulement si :* $\{x \in U : c_i[x] \leq k\} = \{x \in U : c_j[x] \leq k\}, \forall i \neq j$.

Une frontière est donc un entier k tel que l'ensemble des éléments constituant le top-k est identique dans tous les classements consensuels optimaux.

3.6.2 Graphe robuste des éléments G_r et première condition suffisante pour le calcul de frontières

Dans la Section 3.5 de ce même chapitre, nous avons utilisé le graphe G_e afin de concevoir un algorithme de calcul de classements consensuels basé sur une technique de partitionnement. Les CFCs de G_e ordonnées selon un tri topologique forment une partition ordonnée des éléments à trier telle qu'il existe au moins un classement consensuel optimal respectant la partition obtenue. Dans cette section, nous définissions un second graphe étroitement lié au premier. La différence majeure est que les CFCs de ce nouveau graphe, ordonnées selon un tri topologique, forment une partition ordonnée des éléments à trier telle que tous les classements consensuels optimaux respectent la partition obtenue.

Définition 3.6.2 (Graphe robuste des éléments G_r)**.** *Soit* $G_r = (V_r, A_r)$ *le graphe orienté tel que :*

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

Arcs dans le graphe robuste des éléments. Dans G_r, il **n'y a pas** d'arc de x à y si et seulement si $before(y, x)$ est l'*unique minimum* pour la paire $\{x, y\}$ (comparé à $before(y, x)$ et $tied(x, y)$). En d'autres termes, pour une paire $\{x, y\}$, nous pouvons distinguer trois cas :

- cas 1 : $before(x, y)$ est l'unique minimum. Il y a alors un arc de x à y et aucun arc de y à x.
- cas 2 : $before(y, x)$ est l'unique minimum. Il y a alors un arc de y à x et aucun arc de x à y.
- cas 3 : si on ne se trouve ni dans le cas 1 ni dans le cas 2, alors il y a à la fois un arc de x à y et un arc de y à x.

La Table des arcs de G_r (Table 3.5) récapitule les différentes possibilités menant à un arc de x vers y ou de y vers x dans le graphe G_r. Les arcs de G_e sont également rappelés afin de mettre en parallèle les arcs de G_e et ceux de G_r.

TABLE 3.5 – Relation entre la position de min(x,y) et les arcs dans G_e et G_r

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

Remarque. *La Table des arcs de* G_r *(Table 3.5) montre que l'ensemble des arcs de* G_e *est inclus dans l'ensemble des arcs de* G_r.

Différences entre les arcs de G_r et ceux de G_e.

Intéressons-nous maintenant aux différences entre les arcs de G_r et ceux de G_e en reprenant notre exemple Table 3.1 et sa matrice des coûts par paires associée en Table 3.2. On peut voir que le coût de mettre F avant I est identique au coût de mettre I avant F qui est lui-même inférieur au coût de mettre F et I à égalité dans le classement consensuel. Pour le graphe G_e, cela se traduit par une absence d'arc de I à F et de F à I. En effet, l'objectif du graphe G_e est de partitionner en un maximum de sous-problèmes tout en conservant la garantie d'être sur la piste d'un classement consensuel optimal. Vis-à-vis de cet objectif, il y a une indifférence entre le fait de mettre I avant F ou F avant I dans le classement consensuel. Cette indifférence se matérialise par une absence totale d'arc entre les deux éléments. A l'inverse, le graphe robuste G_r doit représenter les points communs entre tous les classements

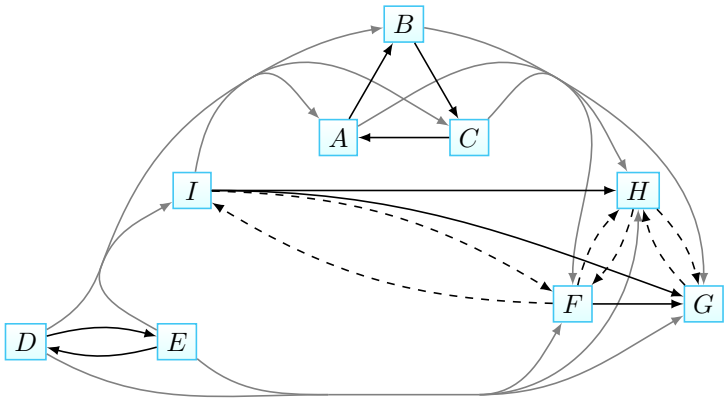


FIGURE 3.3 – Graphe robuste des éléments G_r pour l'exemple de la Table 3.1. Les arcs gris avec sources/cibles multiples indiquent la présence d'un arc de chaque sommet de chaque source vers chaque sommet de chaque cible (par exemple, il y a un arc depuis chaque sommet de (D, E) vers chaque autre sommet). Les arcs en pointillés représentent les arcs de G_r qui ne sont pas présents dans G_e.

consensuels optimaux. L'éventualité de voir F devant I ou I devant F dans un classement consensuel optimal fait que dans ce graphe, les deux éléments sont liés par un arc aller-retour symbolisant l'absence d'une "relation avant" qui serait plus légitime que les autres ordres relatifs possibles. Ainsi, dans G_r, pour une paire d'éléments (x, y), le seul moyen qu'il n'y ait pas d'arc aller-retour est que le coût de mettre x avant y ou y avant x soit l'unique minimum pour cette paire. Si le coût le plus faible est de les mettre à égalité comme pour la paire (D, E) ou qu'il y a une indifférence entre deux positions relatives qui ont le même coût minimal comme pour la paire (F, I), alors on a un arc aller-retour.

Graphe des composantes fortement connexes de G_r et frontières

Considérons maintenant G_r^c le graphe des CFC de G_r. Ce graphe a des caractéristiques remarquables. En effet, G_r^c est un DAG complet puisque pour toute paire de sommets (x, y) de G_r il y a forcément un arc de x à y ou de y a x (ou les deux). Une conséquence directe est que G_r^c a un unique tri topologique.

La Figure 3.4 représente G_r^c pour l'exemple de la Table 3.1.

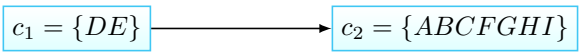


FIGURE 3.4 – G_r^c, le graphe des composantes fortement connexes de G_r.

Intérêt de G_r^c. Nous donnons maintenant une intuition de la raison pour laquelle G_r^c apporte une information complémentaire à G^c. Prenons pour cela deux sommets de G_r^c à savoir deux CFCs distinctes de G_r. Comme par hypothèse ce sont deux CFCs distinctes, soit il n'y a aucun arc de la première composante fortement connexe vers la seconde, soit il n'y a aucun arc de la seconde composante fortement connexe vers la première. Si on regarde

le graphe G_r^c Figure 3.4 ainsi que le graphe G_r Figure 3.3, on observe qu'il n'y a aucun arc de la composante $\{A, B, C, F, G, H, I\}$ vers la composante $\{D, E\}$. D'après la Table 3.5, on peut en conclure qu'il est strictement moins coûteux de mettre n'importe quel élément de $\{D, E\}$ avant n'importe quel élément de $\{A, B, C, F, G, H, I\}$ (par rapport au coût de le mettre après ou à égalité) dans le classement consensuel. Cette observation nous permet d'aboutir à la *propriété fondamentale de* G_r^c à l'origine de frontières.

Propriété 3.6.1 (Propriété fondamentale de G_r^c)**.** *Tous les classements consensuels optimaux respectent l'unique tri topologique* $\mathscr{T}_r = [T_1, ..., T_k]$ *de* G_r^c*.*

Démonstration : Considérons un classement consensuel c qui ne respecte pas $\mathscr{T}_r$ et montrons que c n'est pas optimal. A partir de c, nous construisons un classement consensuel c' comme suit : nous prenons d'abord les éléments de T_1 dans le même ordre relatif que dans c, puis nous ajoutons les éléments de T_2 dans le même ordre relatif que dans c, et nous répétons cette opération pour les éléments de T_3 jusqu'à T_k. Par construction, c' respecte $\mathscr{T}$, donc c' est différent de c. Maintenant, comparons $M(c, R)$ et $M(c', R)$ en utilisant l'équation 3.4. Chaque paire d'éléments (x, y) telle que x et y sont dans le même ordre relatif dans c et c' induit le même coût pour c et c'. Considérons maintenant les paires (x, y) telles que x et y ne sont pas dans le même ordre relatif dans c et c' (il existe au moins une telle paire puisque c ne respecte pas $\mathscr{T}$). Par construction de c', x et y sont dans différents groupes de $\mathscr{T}$. Autrement dit, x et y sont dans deux CFCs différentes de G_r. Posons $x \in T_i$ et $y \in T_j$ avec $i < j$. Comme T_i est avant T_j, il n'y a pas d'arc de y vers x dans G_r. Par construction de G_r, on peut conclure que $before(x, y) = min(x, y)$, $before(y, x) > min(x, y)$ et $tied(x, y) > min(x, y)$. En conséquence, $M(c, r) > M(c', R)$ c'est-à-dire que c ne peut pas être un classement consensuel optimal.

Illustration 3.6.1. *Revenons à la Figure 3.4 représentant* G_r^c *pour notre exemple. On peut conclure grâce à la propriété fondamentale de* G_r^c *que dans tous les classements consensuels optimaux, les gènes* D *et* E *sont positionnés avant tous les autres gènes. On a donc une frontière en position 2.*

Le graphe G_r^c nous permet donc de calculer des frontières mettant ainsi en avant des points communs entre tous les classements consensuels optimaux sans avoir à les calculer.

3.6.3 Combiner G_e et G_r pour trouver plus de frontières

Cette sous-section introduit l'algorithme ParFront qui combine l'information apportée par le graphe des éléments G_e et le graphe robuste des éléments G_r afin de calculer un ensemble de frontières incluant celles obtenues par G_r uniquement.

Notion d'arc robuste

L'objectif de l'algorithme ParFront est de calculer comme précédemment une partition ordonnée $P = [P_1, P_2, ..., P_k]$ de l'ensemble des éléments à trier telle que tous les classements consensuels optimaux respectent P. Nous montrons également dans cette sous-section que la partition obtenue grâce à ParFront est en fait un raffinement de la partition $\mathscr{T}$ présentée dans la Propriété 3.6.1 (pour toute paire d'éléments (x, y) telle que $x \in \mathscr{T}[i]$, $y \in \mathscr{T}[j]$ avec $i < j$, il existe $i' < j'$ tels que $x \in P_{i'}$ et $y \in P_{j'}$).

Reprenons le graphe des éléments G_r et la Table des arcs de G_r (Table 3.5) qui fait le lien entre les arcs de G_r et les valeurs de $before(x, y)$, $before(y, x)$, $tied(x, y)$ et $min(x, y)$. Le seul moyen pour qu'il y ait un arc de x à y sans qu'il n'y ait d'arc de y à x est que le coût de mettre x avant y dans le classement consensuel soit l'unique minimum pour la paire (x, y). Un tel arc de x vers y est informatif dans la mesure où si on ne considère que cette paire, il n'y a aucune ambiguïté sur le fait que x doit être avant y. Ces arcs seront appelés *arcs robustes*.

Définition 3.6.3 (Arc robuste)**.** *Soit $G_r = (V_r, A_r)$ le graphe robuste des éléments. On dira que l'arc $(x, y) \in A_r$ est robuste si et seulement si $(y, x) \notin A_r$.*

Autrement dit, (x, y) est un arc robuste si et seulement si les trois conditions suivantes sont réunies :

- $before(x, y) = min(x, y)$
- $before(y, x) > min(x, y)$
- $tied(x, y) > min(x, y)$

Il est important de noter que tous les arcs robustes sont des arcs de G_e également.

Nous allons maintenant utiliser le graphe G_e combiné à la notion d'arc robuste afin de calculer une partition P raffinant la partition $\mathscr{T}_r$ de la Propriété 3.6.1 permettant ainsi de calculer un ensemble de frontières incluant celles obtenues par $\mathscr{T}_r$.

Exemple d'arcs robustes dans G_e. Presque tous les arcs du graphe G_e représenté en Figure 3.1 sont robustes. Les exceptions sont les arcs (D, E) et (E, D). En effet, pour ces derniers, l'ordre relatif le moins coûteux est l'égalité.

Notation. *Dans la suite de ce chapitre, l'ensemble des arcs robustes sera noté $\mathscr{R}_e$. Les arcs qui ne sont pas $\mathscr{R}_e$ seront appelés arcs non robustes.*

Théorème des arcs robustes

Le graphe G_e dont on distingue les arcs robustes des arcs non robustes fournit une condition suffisante pour qu'une partition P soit respectée par l'ensemble des solutions optimales. Cette condition suffisante est donnée par le théorème ci-dessous.

Théorème 3.6.1 (Théorème des arcs robustes)**.** *Soient $G_e = (V_e, A_e)$ le graphe des éléments et $P = [P_1, P_2, ..., P_k]$ une partition ordonnée de V_e telle que*

- *[1.]* $\forall i < j$, $\forall x \in P_i$, $\forall y \in P_j$, $(y,x) \notin A_e$, *et*
- *[2.]* $\forall i$, $\forall x \in P_i$, $\forall y \in P_{i+1}$, $(x,y) \in \mathscr{R}_e$.

Alors tous les classements consensuels optimaux respectent P.

Démonstration :

Considérons un classement consensuel c qui ne respecte pas P et montrons que c n'est pas optimal. À partir de c, nous construisons un classement consensuel c' de la manière suivante : nous prenons dans un premier temps les éléments de P_1 dans le même ordre relatif que dans c, puis nous ajoutons les éléments de P_2 dans le même ordre relatif que dans c. Nous répétons l'opération pour les éléments de P_3, ..., jusqu'à P_k. Par construction, c' respecte P, donc c' est différent de c. Maintenant, comparons $M(c,R)$ et $M(c',R)$ en utilisant l'équation 3.4 page 60. Chaque paire d'éléments (x,y) telle que x et y sont dans le même ordre relatif dans c et c' induit le même coût pour c et c'.

Considérons maintenant les paires (x,y) telles que x et y ne sont pas dans le même ordre relatif dans c et c' (il y a au moins une telle paire puisque c ne respecte pas P contrairement à c'). Par construction de c', x et y sont dans différents groupes de P. Supposons $x \in P_i$ et $y \in P_j$ avec $i < j$. Alors par hypothèse sur P, il n'y a pas d'arc de y à x dans G_e, c'est-à-dire $before(x,y) = min(x,y)$. Par conséquent, le coût induit par (x,y) est inférieur pour c' que pour c.

De plus, puisque $y \in P_j$ est avant $x \in P_i$ dans c, il existe $i \leq k < j$ et $z \in P_k$, $z' \in P_{k+1}$ tel que z' soit avant z dans c. Mais par construction de c', z est avant z' dans c'. Par les hypothèses du théorème, $(z,z') \in \mathscr{R}_e$, donc $before(z,z')$ est l'unique minimum pour la paire (z,z'). On en déduit que $\overline{M}_{c'}(z,z') < \overline{M}_c(z,z')$.

Au final, on a $\overline{M}_{c'}(x,y) \leq \overline{M}_c(x,y)$ pour chaque paire (x,y) et il y a une inégalité stricte pour au moins une paire. Nous pouvons donc conclure que $M(c,R) > M(c',R)$, c'est-à-dire que c n'est pas optimal.

Rappelons que nous avons défini une frontière comme un entier k tel que l'ensemble des k premiers éléments est le même dans tous les classements consensuels optimaux. Une partition ordonnée $P = [P_1, P_2, ..., P_\ell]$ satisfaisant les conditions du théorème des arcs robustes (Théorème 3.6.1) permet d'obtenir un nombre de frontières correspondant au nombre de parties de P (les frontières sont les sommes cumulées des tailles des parties de P).

L'objectif est maintenant de trouver une partition avec autant de parties que possible et satisfaisant les conditions du théorème des arcs robustes.

Notons qu'il n'est pas possible de trouver une partition satisfaisant à la fois ces conditions et ayant plus de parties que le nombre de composantes fortement connexes de G_e. En effet, l'item 1. du théorème des arcs robustes est satisfait si et seulement si un tri topologique de G^c correspond à une partition plus fine que la partition P (chaque partie de P est l'union de composantes fortement connexes de G_e ordonnées selon un tri topologique).

Pour trouver une partition avec autant de parties que possible satisfaisant aux conditions du Théorème 3.6.1,

nous commençons par un tri topologique de G^c (satisfaisant ainsi la condition 1), et fusionnons à plusieurs reprises des parties lorsque la condition 2 n'est pas satisfaite. C'est ce que fait l'algorithme ParFront ci-dessous.

ParFront, algorithme de calcul de frontières

Une implémentation de ParFront est donnée en Algorithme 2.

Algorithme 2 : ParFront : retourne les frontières obtenues par le Théorème 3.6.1.

Input : $T = [T_1, ..., T_k]$: liste des composantes fortement connexes de G_e selon un tri topologique, Re : ensemble des arcs robustes de G_e

begin

```
/* Partie 1 : calcul de la partition ordonnée P                    */

P ← T /* P est une copie de T. Ainsi, P satisfait la condition 1.  */
i ← 1
while i < size(P) do
    /* Vérification de la condition 2 entre P[i] et P[i+1]         */
    if ∃x ∈ P[i], y ∈ P[i+1], (x, y) ∉ Re then
        P[i] ← P[i] ∪ P[i+1] /* fusion de P[i] et P[i+1]           */
        supprime P[i+1]
        i ← max(i−1, 1) /* retour en arrière d'un pas pour i       */
    else
        i ← i+1 /* avance d'un pas                                 */

/* Partie 2: utilisation de P pour calculer les frontières         */

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

Result : Liste d'entiers (position des frontières)

Remarque. *L'Algorithme ParFront peut être utilisé de deux manières différentes. Tout d'abord, l'utilisateur peut vouloir obtenir un classement grossier des éléments. Dans ce cas, ParFront peut être exécuté seul pour fournir un classement grossier robuste (la partition ordonnée P). Alternativement, l'utilisateur peut vouloir obtenir un classement total «sensible à la robustesse» des éléments. Dans ce cas, ParFront est exécuté après ParCons pour fournir un classement consensuel avec des frontières solides.*

Dans ce qui suit, nous montrons que le résultat de ParFront est optimal dans le sens où il fournit l'unique partition qui a le plus grand nombre de parties parmi celles satisfaisant les hypothèses du théorème.

Dans un premier temps, nous rappelons qu'une partition ordonnée $P = [P_1, \ldots, P_k]$ est un *raffinement* d'une autre partition ordonnée $P' = [P'_1, \ldots, P'_\ell]$ si toute partie de P' est une union de parties consécutives de P.

Propriété 3.6.2. *La partition donnée par l'Algorithme 2 satisfait les hypothèses du théorème des arcs robustes (Théorème 3.6.1). De plus, cette partition est un raffinement de toutes les autres partitions qui satisfont ces hypothèses.*

Notons que cette propriété implique que le résultat de ParFront est indépendant du tri topologique donné en entrée (en effet si P raffine P' et P' raffine P, alors $P = P'$).

Démonstration : On appelle *partition-frontière* une partition ordonnée P satisfaisant les conditions du théorème des arcs robustes (Théorème 3.6.1).

La preuve est en deux parties : premièrement, nous prouvons que l'algorithme ParFront donne une partition-frontière, puis nous affirmons que c'est un raffinement de toute partition-frontière.

Pour la première partie, nous prouvons d'abord la condition 1 : à aucun moment de l'algorithme il n'y a un arc entre un élément de $P[j]$ et un élément de $P[i]$, avec $j < i$. En effet, ceci est vrai à l'initialisation de P (puisque nous partons d'un tri topologique), et n'est pas invalidé par la fusion de parties consécutives. De plus, pendant la boucle *while*, la condition 2 est satisfaite pour chaque paire $P[j], P[j+1]$, où j est plus petit que la valeur courante de i. En quittant la boucle quand $i = size(P)$, la condition 2 est alors satisfaite pour chaque paire $P[i]$, $P[i+1]$, donc ParFront produit une partition-frontière.

Pour la seconde partie, notons tout d'abord qu'à aucun moment de l'algorithme il ne peut y avoir de chemin d'un élément de $P[j]$ vers un élément de $P[i]$, avec $i < j$ (c'est une conséquence de la condition 1). Nous appelons cette propriété la *propriété de chemin*. Nous considérons maintenant une partition frontière P', et montrons par récurrence qu'en tout point de l'algorithme, P est un raffinement de P'. En effet, P commence par T et T est un raffinement de P' (sinon P' violerait la condition 1). Nous montrons maintenant qu'à chaque fois que P est modifié, le résultat reste un raffinement de P'. Si P est modifié, cela signifie qu'il y a des i et des (x, y) tels que $x \in P[i], y \in P[i+1], (x, y) \notin \mathscr{R}_e$. Puisque P est un raffinement de P', il existe des (j, j') tels que $P[i] \subseteq P'[j]$ et $P[i+1] \subseteq P'[j']$. Notre but est de montrer que $j = j'$: lors de la fusion de $P[i]$ et $P[i+1]$, le résultat est toujours un raffinement de P'.

Dans l'objectif d'aboutir à une contradiction, supposons que $j \neq j'$. Nous avons $j < j'$ (sinon, puisque P' satisfait la condition 2, il y aurait un chemin de y à x, ce qui est exclu par la propriété de chemin). Si $j' = j+1$, alors P' viole la condition 2 (en effet nous avons supposé $(x, y) \notin \mathscr{R}_e$). Donc $j+1 < j'$. On considère $z \in P'[j+1]$: il existe un chemin robuste (en fait un arc) de x à z et un chemin robuste de z à y, donc à partir de la propriété de chemin, z ne peut pas être dans une partie strictement avant x dans P (c'est-à-dire pas avant $P[i]$), ni dans une partie strictement après y (pas après $P[i+1]$). Donc z est soit dans $P[i]$ soit dans $P[i+1]$, c'est à dire soit dans la même partie de P que x soit dans la même partie de P que y. Cela contredit l'hypothèse de récurrence, car au début de cette étape P est un raffinement de P' et z n'est pas dans la même partie de P' que x ou y. Nous avons obtenu la contradiction souhaitée, montrant que lors de la fusion de $P[i]$

et $P[i+1]$, le résultat est toujours un raffinement de P'. On en conclut que P reste un raffinement de P' tout au long de l'algorithme.

Au final, l'algorithme produit une partition-frontière qui est un raffinement de chaque autre partition frontière : ainsi, il donne une partition frontière de taille maximale, qui est unique.

Complexité de l'algorithme ParFront

Pour rappel, ParFront part du tri topologique des composantes fortement connexes de G^c et fusionne ou non des CFC contiguës.

On suppose qu'on a déjà calculé la matrice des coûts et le tri topologique de G^c (ce qui est bien le cas si on a d'abord lancé l'algorithme ParCons). Pour rappel, la complexité de ces pré-calculs est $O(|R| * |U|^2)$.

L'algorithme ParFront fusionne deux CFCs contiguës si et seulement si il existe x appartenant à la première CFC et y appartenant à la CFC suivante tels que (x, y) n'est pas un arc robuste.

Chaque paire d'éléments de U est inspectée au pire une fois, et savoir si (x, y) est un arc robuste est de complexité constante. Chaque paire d'éléments de U est effectivement inspectée une fois dans le cas où toutes les CFCs sont de taille 1 et que pour tout i, on se rend compte que la CFC i fusionne avec la CFC $i+1$ à la dernière vérification d'existence d'arc robuste. On remarque également que le nombre de déplacements d'éléments dans une autre CFC ne peut pas dépasser le nombre de vérifications d'arcs robustes. La complexité au pire (sans les pré-calculs) est au final de $O(|U|^2)$. Si on intègre les pré-calculs, la complexité est alors $O(|R| * |U|^2)$.

3.6.4 Comparaison avec l'état de l'art

Bien que nous n'ayons connaissance d'aucun travail ayant pour but d'aider les utilisateurs à repérer des domaines robustes au sein des classements consensuels optimaux, certains travaux précédents, notamment [19] et [83], ont introduit des algorithmes pour partitionner les éléments à trier de sorte que tous les classements consensuels optimaux respectent la partition obtenue. Ces travaux ont ainsi défini des concepts qui peuvent être repensés en termes de frontières.

Dans cette section, nous montrons que les frontières calculées par ParFront incluent toutes celles qui peuvent être trouvée grâce aux travaux de [19] et [83] décrits dans notre état de l'art (voir Section 2.2) et repris ci-après.

Candidats propres

Betzler et al. [19] ont défini le concept *d'élément propre en proportion 0.75)* (*non-dirty candidates for 0.75*). Il s'agit d'un élément x tel que pour tout élément $y \neq x$, soit x est devant y dans au moins 75 % des classements d'entrée, soit y est avant x dans au moins 75% des classements d'entrée. Betzler et al. ont prouvé que :

- tous les éléments placés avant x dans au moins 75 % des classements sont placés avant x dans tous les classements consensuels optimaux.
- tous les éléments placés après x dans au moins 75 % des classements sont placés après x dans tous les classements consensuels optimaux.

Ce théorème a été démontré dans le cas où les classements en entrée sont des permutations d'un même ensemble mais il est facile de démontrer que ce théorème reste vrai pour les classements complets avec égalité en considérant le score de Kemeny généralisé avec $p = 1$ (voir Définition 1.3.2).

Le fait de trouver un élément propre aboutit donc à une partition ordonnée des éléments à trier $\mathscr{B} = [B_1, B_2, B_3]$ constituée des trois parties suivantes :

- B_1 est l'ensemble de tous les éléments placés avant x dans au moins 75% des classements d'entrée
- $B_2 = \{x\}$
- B_3 est l'ensemble de tous les éléments placés après x dans au moins 75% des classements d'entrée

Nous considérons donc la procédure suivante : s'il y a un élément propre x, on calcule dans un premier temps B_1 et B_3 puis on cherche récursivement les candidats propres de B_1 et B_3. En effet, un élément de B_1 peut être propre pour B_1 même s'il ne l'est pas pour $B_1 \cup B_2 \cup B_3$.

Cette procédure est déterministe : l'ordre dans lequel on sélectionne les candidats propres n'a pas d'incidence sur la partition finale. En effet, on remarque que si un élément est un élément propre dans $B_1 \cup B_2 \cup B_3$, alors il reste un élément propre dans B_1 ou B_3.

On note $\mathscr{B} = [B_1, B_2, ..., B_b]$ la partition obtenue après la procédure récursive.

Propriété 3.6.3. *Toutes les frontières données par la partition $\mathscr{B}$ (éléments propres) sont également données par la partition P calculée par ParFront (Algorithme 2).*

Démonstration : La Propriété 3.6.2 indique que la partition P obtenue par ParFront raffine toute partition satisfaisant les hypothèses du Théorème 3.6.1. Il suffit donc de prouver que $\mathscr{B}$ respecte les hypothèses du théorème des arcs robustes (Théorème 3.6.1).

Premièrement, prouvons que $\mathscr{B}$ respecte la condition 1 du Théorème 3.6.1. Soit $i < j$, $x \in B_i$, $y \in B_j$. Soit z l' élément propre qui sépare x et y (z peut être x ou y). Puisque $i < j$, nous savons que soit $x = z$, soit x est avant z dans au moins 75 % des classements. De même, nous savons que soit $y = z$, soit z est avant y dans au moins 75% des classements. Dans tous les cas, nous avons x avant y dans au moins 50% des classements, donc $before(x, y) = min(x, y)$, c'est-à-dire $(y, x) \notin A_e$. Ainsi $\mathscr{B}$ respecte l'hypothèse 1 du théorème des arcs robustes.

Deuxièmement, prouvons que $\mathscr{B}$ respecte la condition 2 du Théorème 3.6.1. Soit $i < b$, $x \in B_i$, $y \in B_{i+1}$. Par construction de B, soit x est un élément propre par rapport à $B_{i-1} \cup B_i \cup B_{i+1}$, soit y est un élément propre

par rapport à $B_i \cup B_{i+1} \cup B_{i+2}$. Dans les deux cas, x est avant y dans 75 % des classements, soit $(x, y) \in \mathscr{R}_e$ donc $\mathscr{B}$ respecte la condition 2 du théorème des arcs robustes. On en conclut que P raffine $\mathscr{B}$.

Critère de Condorcet étendu

Truchon prouve dans [83] que pour un nombre impair de classements complets (pouvant avoir des égalités), tous les classements consensuels optimaux respectent toute partition ordonnée $\mathscr{X} = [X_1, X_2, ..., X_k]$ tel que $\forall i < j$, $\forall x \in X_i$, $\forall y \in X_j$, x est avant y dans une majorité de classements. Ce théorème est connu sous le nom de *Critère de Condorcet étendu*. Notons que Truchon fixe $p = 1$ pour le coût de créer ou rompre une égalité.

On peut étendre ce résultat pour qu'il reste vrai pour un nombre pair de classements.

La condition suivante est donc vraie : tout classement consensuel optimal respecte toute partition ordonnée $\mathscr{X} = [X_1, X_2, ..., X_k]$ telle que $\forall i < j$, $\forall x \in X_i$, $\forall y \in X_j$, x est avant y dans une stricte majorité de classements.

Remarque. *Lorsque le nombre de classements est impair, pour toute paire d'éléments (x, y), on a forcément x avant y dans une stricte majorité de classements ou y avant x dans une stricte majorité de classements. Truchon n'a donc pas besoin de préciser que la relation de préférence soit stricte. Lorsque nous étendons son théorème pour un nombre pair de classements, il est obligatoire de préciser que les relations de préférence doivent être strictes.*

On considère maintenant que $\mathscr{X}$ est la partition maximale (maximale en nombre de parties) respectant la condition ci-dessus.

Propriété 3.6.4. *Toutes les frontières données par la partition $\mathscr{X}$ (Critère de Condorcet étendu) sont également données par la partition P calculée par ParFront (Algorithme 2).*

Démonstration : Prouvons que $\mathscr{X}$ respecte les deux conditions du théorème des arcs robustes (Théorème 3.6.1). Tout d'abord, posons $i < j$, $x \in X_i$, $y \in X_j$. Par hypothèse, x est avant y dans une stricte majorité de classements. On obtient $before(x, y) < before(y, x)$ et $before(x, y) < tied(x, y)$, soit $(y, x) \notin A_e$ et en conséquence $\mathscr{X}$ respecte l'hypothèse 1 du théorème des arcs robustes. De plus, comme $before(x, y)$ est le minimum unique, $(x, y) \in \mathscr{R}_e$. Ainsi $\mathscr{X}$ respecte l'hypothèse 2 du théorème des arcs robustes. On peut conclure que P raffine $\mathscr{X}$.

Graphe robuste des éléments

Considérons maintenant le graphe robuste des éléments (Définition 3.6.2). La Propriété 3.6.1 indique que tous les classements consensuels optimaux respectent l'unique tri topologique $\mathscr{T} = [T_1, T_2, ..., T_k]$ du graphe des composantes fortement connexes de G_r.

Propriété 3.6.5. *Toutes les frontières données par la partition $\mathscr{T}$ (composantes fortement connexes du graphe robuste) sont également données par la partition P calculée par ParFront (Algorithme 2).*

Démonstration : Nous montrons que $\mathscr{T}$ respecte les hypothèses du théorème des arcs robustes. Tout d'abord, soit $i < j$, $x \in X_i$, $y \in X_j$. Comme $\mathscr{T}$ est un tri topologique du graphe des composantes fortement connexes de G_r, x et y sont dans différentes composantes fortement connexes de G_r et nous avons $(y, x) \notin A_r$. Par définition de A_r, puisque $(y, x) \notin A_r$ alors nécessairement $before(x, y)$ est l'unique minimum pour la paire (x, y), c'est-à-dire $(x, y) \in \mathscr{R}_e$. Nous pouvons conclure que $\mathscr{T}$ respecte l'hypothèse 2 du théorème des arcs robustes. De plus, puisque $before(x, y) = min(x, y)$, on a $(y, x) \notin A_e$. On obtient donc que $\mathscr{T}$ respecte l'hypothèse 1 du théorème des arcs robustes. Au final, P raffine $\mathscr{T}$.

Nous avons considéré des travaux ayant défini des concepts qui peuvent être repensés en termes de frontières et prouvé que si une frontière est trouvée par une des méthodes citées ci-dessus, alors elle est également trouvée par ParFront. A notre connaissance, ParFront est donc actuellement l'approche la plus générale capable d'identifier des régions robustes dans le classement consensuel.

En complément de l'algorithme ParCons permettant de calculer un classement consensuel de qualité en un temps raisonnable, l'algorithme ParFront est capable de calculer des frontières mettant ainsi en avant certaines zones du classement consensuel qui peuvent être considérées comme particulièrement robustes.

3.7 Nouvelle version de ConQuR-Bio

Les algorithmes ParCons et ParFront décrits dans les sections précédentes ont été implémentés dans l'outil en ligne ConQuR-BioV2 que nous décrivons dans cette section. Cet outil utilisé par la communauté des Sciences de la Vie est disponible à l'adresse suivante : http://conqur-bio.lri.fr.

Nous commençons par présenter le logiciel ConQuR-Bio développé par Bryan Brancotte et publié dans [24]. Nous mettons ensuite en avant les changements apportés au logiciel ayant abouti à la publication de ConQuR-BioV2 [8].

3.7.1 Le logiciel ConQuR-Bio

Le but de ConQuR-Bio (initialement introduit dans [24]) est de fournir une liste de gènes associés à une maladie (exprimée sous la forme de mot-clé par les utilisateurs). ConQuR-Bio exploite le fait que les noms de maladies peuvent avoir différents synonymes. Chaque synonyme est associé à une liste différente de gènes qui est récupérée

par ConQuR-Bio afin de fournir aux utilisateurs un classement consensuel des gènes d'intérêt pour la maladie considérée.

Description de l'architecture de ConQuR-Bio. L'architecture de ConQuR-Bio, décrite dans la Figure 3.5, est composée de trois modules principaux.

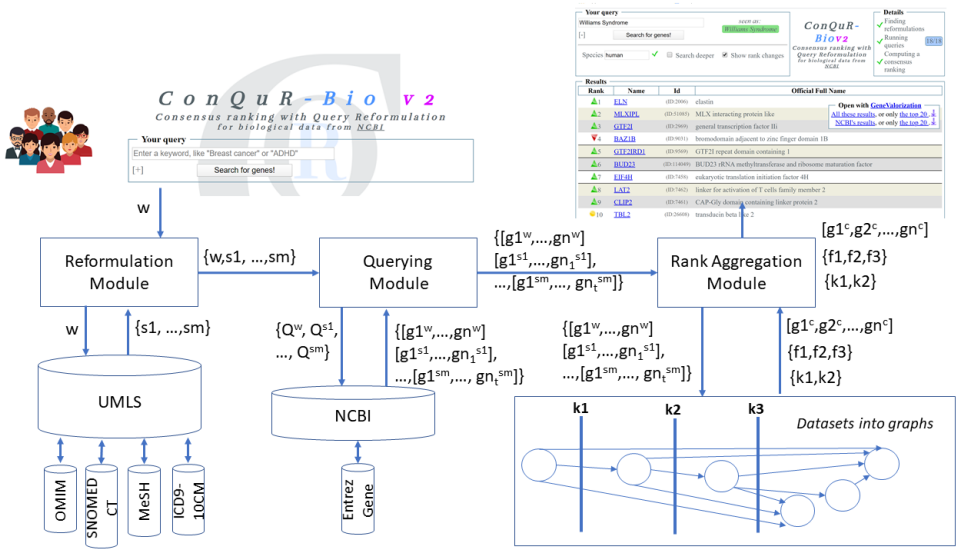


FIGURE 3.5 – Architecture de l'outil ConQuR-Bio

Le premier module - **le module de reformulation** - prend comme entrée w, le mot-clé utilisateur (par exemple "breast cancer") et génère un ensemble de synonymes de w ($\{s_1, ..., s_m\}$). Les synonymes sont générés grâce à une utilisation automatique de l'UMLS Metathesaurus [22] qui interroge cinq bases de données décrites ci-dessous et sélectionnées avec nos collaborateurs biologistes pour fournir des synonymes pour les noms de maladies.

- MeSH (Medical Subject Headings) [65] contient un grand ensemble de mots-clés médicaux car il a été conçu pour indexer les mots-clés associés aux publications médicales dans PubMed.
- SNOMED CT (SNOMED Clinical Terms) [81] est un ensemble de termes médicaux fournissant des synonymes et des définitions utilisés dans la littérature et les rapports cliniques.
- Les deux dernières versions de la Classification Internationale des Maladies (CIM-9-CM et CIM-10-CM) [75, 76], ont été développées par l'Organisation Mondiale de la Santé. La CIM-9-CM est le système officiel d'attribution de codes aux diagnostics et aux procédures d'utilisation des hôpitaux (aux États-Unis et en Europe). La CIM-10-CM contient des codes supplémentaires pour les causes de mortalité.
- L'héritage mendélien en ligne chez l'humain (OMIM) [52] contient un catalogue très riche de toutes les maladies génétiques connues du génome humain.

Le deuxième module - **le module d'interrogation** - prend le mot-clé utilisateur et la liste des synonymes fournis par le module de reformulation et envoie une requête à la base de données *Gene* du NCBI par mot-clé et syno-

nyme. Pour chaque mot-clé (et synonyme), *Gene* du NCBI fournit la liste de gènes associée. Dans la Figure 3.5, $g1^w, ..., gn^w$ est la liste des gènes associée au mot-clé w. Chaque liste de gènes est classée par le NCBI par ordre décroissant du nombre d'occurrences du mot-clé (ou du synonyme) dans la fiche du gène. En conséquence, le module d'interrogation produit plusieurs listes triées de gènes (une par mot-clé et synonyme).

Le troisième module - le **module d'agrégation de classements** - prend un ensemble de listes de gènes et utilise l'algorithme BioConsert (décrit dans la Sous-section 2.1.3) pour calculer un classement consensuel.

Limites de ConQuR-Bio. Si ConQuR-Bio a été utilisé par plusieurs membres de la communauté des Sciences de la Vie, il souffrait de deux faiblesses majeures : (i) le nombre d'éléments obtenus en réponse à une requête (gènes associés à un mot-clé) augmentait constamment, il y avait donc un besoin crucial d'une approche capable d'évoluer tout en n'utilisant pas systématiquement l'heuristique BioConsert ; (ii) ConQuR-Bio a été utilisé pour fournir de nouvelles pistes de recherche pouvant par la suite être confirmées par des expériences biologiques ; les médecins utilisant cet outil ont exprimé la nécessité d'avoir des indicateurs sur la robustesse associée aux positions des éléments dans le classement consensuel fourni. En effet, BioConsert est une heuristique qui ne repose que sur une méthode de recherche locale. Par essence, elle est incapable de fournir à l'utilisateur des indices sur la qualité et/ou la robustesse des classements consensuels calculés.

3.7.2 ConQuR-BioV2 : prise en compte des retours des biologistes

Modifications apportées. ConQuR-BioV2 apporte une réponse aux deux limites formulées au paragraphe précédent en introduisant deux nouvelles fonctionnalités clés : (i) le module d'agrégation de classements a été entièrement reconstruit afin d'intégrer l'algorithme ParCons et (ii) l'interface utilisateur a été adaptée afin de matérialiser par un trait en gras les différentes frontières calculées par ParFront.

Intérêt de ConQuR-BioV2. Nous considérons ici un cas réel d'utilisation de ConQuR-BioV2, basé sur la recherche de gènes liés au *Syndrome de Williams*, un trouble présentant des manifestations cliniques de diverses natures dont des anomalies cardiaques et un retard mental. Le syndrome de Williams est une maladie rare (environ 1 naissance sur 7500) nommée par plusieurs dénominations parfois très différentes dans la littérature (par exemple, «Beuren Syndrome», « Hypercalcemia Supravalvar Aortic Stenosis »). Selon *Orphanet* (`orphanet.org`), la source d'information de référence sur les maladies rares, huit gènes sont connus pour être également fortement associés à la maladie, à savoir (par ordre alphabétique), BAZ1B, CLIP2, ELN, GTF2I, GTF2IRD1, LIMK1, RCF2 et TBL2.

Dans l'exemple fourni en Figure 3.6, l'utilisateur a considéré le mot-clé *Williams Syndrome* pour lequel 18 synonymes ont été trouvés. La liste ordonnée des gènes et des frontières peut être visualisée par l'utilisateur (les frontières sont matérialisées par des traits gras).

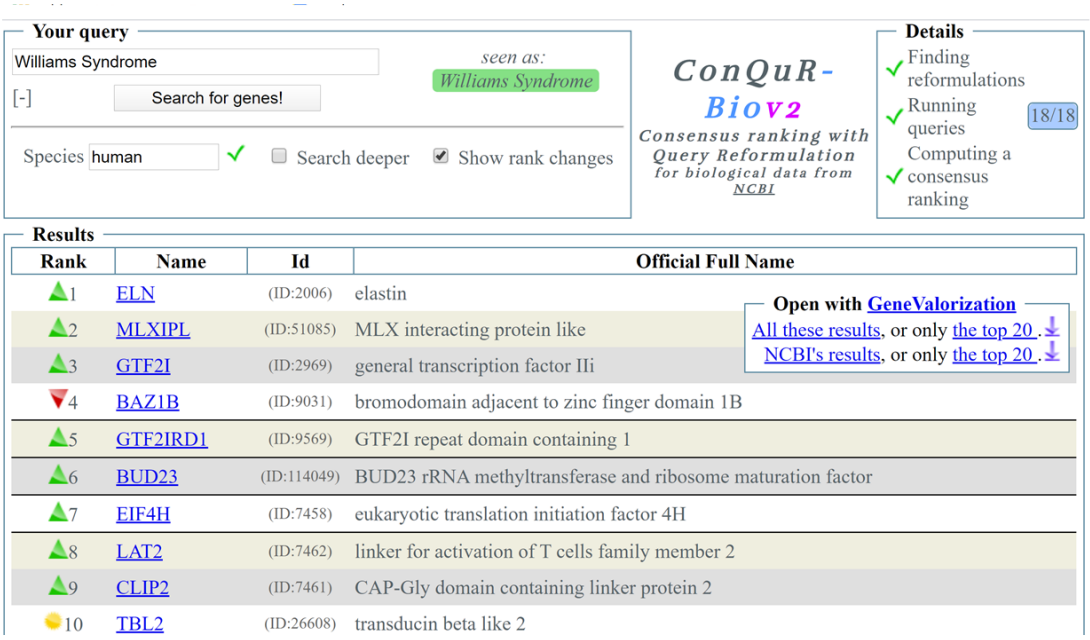


FIGURE 3.6 – Interface de ConQuR-BioV2 et classement consensuel obtenu pour le mot clé (maladie) "Williams Syndrome".

Notons que le nom de la maladie est automatiquement coloré en vert car il est reconnu comme terme MeSH. Ensuite, le module de reformulation a été exécuté et, comme indiqué sur le côté droit de l'interface, a obtenu 18 reformulations de l'UMLS (soit 18 dénominations alternatives). Enfin, le classement consensuel des gènes à considérer est fourni à l'utilisateur. La position de chaque gène dans le classement d'origine de la base de données *Gene* du NCBI, c'est-à-dire le classement fourni par le NCBI avec le mot-clé utilisateur (ici « Williams Syndrome »), est donnée à titre indicatif. Lorsque ConQuR-BioV2 a classé le gène plus haut (respectivement, plus bas) que le NCBI, une flèche verte vers le haut (respectivement une flèche rouge vers le bas) est affichée ; lorsque ConQuR-BioV2 a fourni un gène que NCBI ne fournit pas, un soleil jaune s'affiche (l'affichage de ces informations n'est pas une nouveauté de ConQuR-BioV2).

Un examen plus approfondi des résultats montre deux points très intéressants.

- les 8 gènes particulièrement liés à la maladie selon *Orphanet* ont été renvoyés par ConQuR-BioV2 alors que NCBI n'a pas renvoyé le gène TBL2 ;
- le top-5 (respectivement, top-10) des gènes renvoyés par ConQuR-BioV2 contient 4 (respectivement, 6) gènes également renvoyés par *Orphanet* alors que NCBI ne renvoie que 2 de ces gènes dans son top-10.

Un dernier point à noter est l'information offerte par les frontières : ConQuR-BioV2 fournit aux utilisateurs quatre frontières en position 4, 5, 6 et 7. Ces frontières indiquent que les gènes ELN, MLXIPL, GTF2I, BAZ1B doivent être considérés en priorité, puis GTF2IRD1, puis BUD23 et enfin EIF4H.

L'intégration des algorithmes ParCons et ParFront dans ConQuR-BioV2 permet de fournir aux médecins et

biologistes des informations précieuses quant à la robustesse du classement consensuel renvoyé. Nous observons que le classement consensuel associé à la requête "Williams Syndrome" met en avant plus de gènes considérés par Orphanet comme particulièrement associés à cette maladie que le classement du NCBI ne prenant pas en compte les reformulations. La section suivante est dans la continuité de cette observation : elle a pour objet d'évaluer quantitativement et qualitativement les algorithmes ParCons et ParFront.

3.8 Évaluation quantitative des algorithmes ParCons et ParFront

Cette section est dédiée à l'évaluation (quantitative) de l'efficacité des techniques algorithmiques introduites dans ce chapitre respectivement relatives à (i) l'introduction d'un algorithme de partitionnement ParCons permettant d'optimiser le calcul d'un classement consensuel et à (ii) l'introduction d'une technique de calcul de *frontières* par l'algorithme ParFront mettant en évidence les points communs d'une liste de classements consensuels optimaux.

3.8.1 Jeux de données considérés

Nous considérons ici un ensemble de jeux de données relatifs aux données constituées de classements de gènes associés à des maladies (cas d'utilisation décrit en Sous-section 3.1.1). Chaque jeu de données du Dataset décrit ci-dessous est constitué d'une liste de classements. Chacun d'entre eux correspond aux classements des gènes les plus pertinents vis-à-vis d'un nom de maladie donnée, ou d'un de ses synonymes.

Dataset 1 (reformulations synonymes de maladies)**.** *Le Dataset 1 est constitué d'un ensemble de 1 354 noms de maladies ayant au moins 3 reformulations synonymes dans la terminologie MeSH (c'est-à-dire des noms de maladie pour lesquels il existait au moins 3 noms synonymes de la maladie) et étant associés (toutes refomulations confondues) à au moins 100 gènes dans la base de données Gene du NCBI. Sur ces 1 354 noms de maladie, le nombre de reformulations se situe entre 3 et 120 avec une moyenne de 11.5 et le nombre de gènes impliqués se situe entre 100 et 1 557, avec une moyenne de 295. La méthode utilisée pour obtenir les différentes reformulations est détaillée dans la Section 3.7 de ce chapitre.*

Le Dataset 1 nous sert donc à évaluer l'efficacité des algorithmes ParCons et ParFront. Dans ce contexte biologique, c'est le score de Kemeny généralisé à la pseudo-distance (Définition 2.3.6) qui est utilisé [7, 8, 24]. Les algorithmes utilisés ici sont donc les algorithmes décrits dans la Sous-section 4.3.3 qui sont compatibles avec cette adaptation du score de Kemeny pour gérer les données manquantes : KwikSort [3], Copeland Method [36], BioConsert [32], et l'algorithme exact basé sur de la PLNE dans le cadre du partitionnement avec ParCons (l'algorithme exact a été conçu dans un cadre plus général et est décrit au chapitre suivant dans la Sous-section 4.3.1).

L'évaluation des algorithmes ParCons et ParFront est segmentée en cinq expériences publiées dans [8].

Nous avons évalué dans un premier temps la capacité de l'étape de partitionnement à positionner des éléments à leur place définitive dans le classement consensuel, sans l'aide d'algorithmes auxiliaires. Pour rappel, si le fait de mettre à égalité tous les gènes d'un même sous-problème forme un classement consensuel optimal pour le sous-problème considéré, on considère que le partitionnement a classé les gènes sans l'aide d'un algorithme auxiliaire. C'est le cas, en particulier, des sous-problèmes consistant à classer un unique gène.

Nous nous sommes ensuite intéressés à la taille des sous-problèmes nécessitant l'utilisation d'un algorithme auxiliaire afin d'évaluer la capacité à utiliser un algorithme exact plutôt qu'une heuristique. Nous avons également comparé la qualité des heuristiques listées ci-dessus que nous avons généralisées pour prendre en compte le score de Kemeny généralisé à la pseudo-distance, ainsi que leur temps d'exécution. Enfin, nous avons évalué la capacité de l'algorithme ParFront à calculer des frontières pertinentes et comparé le nombre de frontières obtenues avec les autres méthodes de l'état de l'art.

3.8.2 Évaluation de ParCons sur des données biologiques

Les expériences 1 à 3 évaluent la capacité de ParCons à partitionner le problème initial en sous-problèmes, ainsi que la capacité à utiliser un algorithme exact.

Expérience 1 : évaluation de la capacité de ParCons à partitionner. La première expérience consiste à évaluer l'impact du partitionnement dans le calcul du classement consensuel. Les Figures 3.7 et 3.8 montrent qu'une grande quantité de gènes peut être classée par la phase de partitionnement sans avoir besoin d'utiliser un algorithme auxiliaire.

Plus précisément, la Figure 3.7 montre que dans 396 jeux de données, 100 % des gènes sont placés dans le classement consensuel en utilisant uniquement le partitionnement (c'est-à-dire qu'un classement consensuel optimal est trouvé). De plus, dans 665 jeux de données, plus de 80 % des gènes sont placés dans le classement consensuel par le partitionnement uniquement (ce nombre est obtenu en considérant les parties 100 % et 80-99 % de la Figure 3.7, avec 665 = 269 + 396 jeux de données).

Pour obtenir la Figure 3.8, nous avons sommé sur les 1354 jeux de donnés le nombre de gènes mis à leur place définitive dans le classement consensuel par (i) le partitionnement, (ii) un algorithme exact, (iii) une heuristique. Le résultat est le suivant : 67,5 % des gènes sont placés dans le classement consensuel en utilisant uniquement le partitionnement. Pour information, le nombre total de gènes est de 269 564.

Pour classer les éléments qui n'ont pas pu être mis à leur place définitive dans le classement consensuel par le partitionnement (les 32.5 % de gènes restants), il y a deux possibilités.

La première consiste à favoriser, lorsque cela est possible, l'utilisation d'un algorithme exact. L'avantage est la garantie que le classement consensuel calculé pour le sous-problème considéré est optimal. L'inconvénient est que

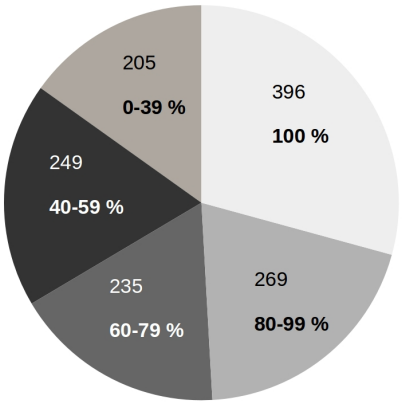


FIGURE 3.7 – Nombre de jeux de données pour lesquels la proportion de gènes placés dans le classement consensuel par le partitionnement uniquement (sans appel à un algorithme auxiliaire) est dans les proportions suivantes : (0-39%, 40-59%, 60-79%, 60-99%, 100%).

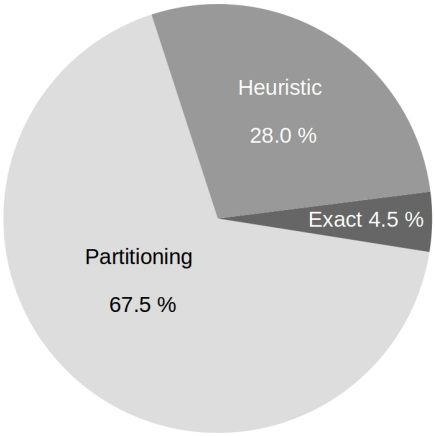


FIGURE 3.8 – Nombre total de gènes placés dans le classement consensuel par le partitionnement, un algorithme exact et une heuristique.

si le nombre de gènes est trop important (la composante connexe du graphe contient trop d'éléments, voir Section 3.5.1), alors le temps de calcul est bien trop élevé.

La deuxième possibilité consiste à utiliser une heuristique. L'inconvénient est ici l'absence d'information quant à la qualité du classement consensuel renvoyé.

L'expérience suivante évalue la capacité à se servir de l'algorithme exact, autrement dit la capacité à partitionner le problème initial en des sous-problèmes de petite taille.

Expérience 2 : capacité à utiliser l'algorithme exact. En pratique, ParCons utilise l'algorithme exact pour calculer un classement consensuel associé à un sous-problème si le nombre de gènes à classer dans le sous-problème est inférieur à 80 (la composante fortement connexe en question contient moins de 80 gènes). Cela garantit que le

classement consensuel final peut être renvoyé à l'utilisateur dans un délai raisonnable (quelques secondes).

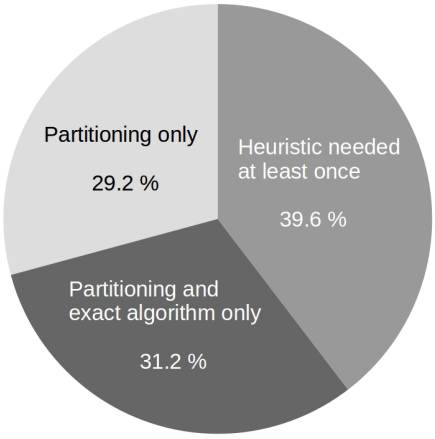


FIGURE 3.9 – Proportion de jeux de données telle que le partitionnement a été utilisé seul (classement consensuel optimal), le partitionnement et un algorithme exact ont été utilisés, sans heuristique (classement consensuel optimal), une heuristique était nécessaire pour au moins un sous-problème (classement consensuel éventuellement non optimal).

La Figure 3.9 présente la proportion de jeux de données pour lesquels le classement consensuel fourni à l'utilisateur a été obtenu en utilisant (i) le partitionnement uniquement (cf Expérience 1), (ii) le partitionnement et algorithme exact (sans utilisation d'une heuristique), (iii) le partitionnement et une heuristique. Notons que (i) et (ii) fournissent un classement consensuel optimal. Grâce à l'algorithme exact, un classement consensuel optimal a été trouvé dans 60,4 % des jeux de données (29,2 + 31,2) c'est-à-dire pour 818 jeux de données. Pour information, un total de 287 531 gènes (représentant 72 % du nombre total de gènes) a pu être positionné par le partitionnement ou un algorithme exact. La Figure 3.9 indique qu'une heuristique doit être utilisée pour 39,6 % des jeux de données, à savoir pour 536 jeux de données.

Déterminer la meilleure heuristique à choisir est donc une question clé. En effet, pour les 39,6 % jeux de données nécessitant une heuristique, il est important de choisir une heuristique dont les résultats donnent de bons résultats en pratique. Cette question est donc considérée dans l'expérience suivante.

Expérience 3 : banc d'essai des heuristiques, avec et sans partitionnement. L'expérience consiste à calculer pour chacun de ces algorithmes le score de Kemeny généralisé à la pseudo-distance du classement consensuel obtenu ainsi que le temps d'exécution pour parvenir au calcul du classement consensuel.

Conditions de l'expérience. Les expériences ont été menées en utilisant un quadruple *dual-core processor Intel Core 2.9GHz* avec 32GB de mémoire vive. Le code source est implémenté en Java 1.8.0. Chaque mesure de temps est précédée d'un temps de chauffe permettant d'assurer que toutes les classes sont déjà chargées dans la JVM. Aucun algorithme n'a utilisé de *multi-threading*.

Afin d'évaluer la qualité des classements consensuels obtenus, nous avons considéré une mesure classique-

ment utilisée dans ce cadre : le *gap* [5, 78]. Cette mesure consiste à normaliser le score de Kemeny associé au classement consensuel c afin de montrer son éloignement vis-à-vis du score de Kemeny qu'on obtiendrait avec un algorithme exact renvoyant un classement consensuel optimal c^*.

Définition 3.8.1. *Soit R une liste de classements et c^* un classement consensuel optimal. Le gap est défini de la manière suivante.*

$gap(c, R) = \frac{M(c,R)}{M(c^*,R)} - 1$

Remarque. *Le gap (qui est une valeur positive ou nulle) vaut 0 si et seulement si le classement consensuel renvoyé est optimal.*

Si un classement consensuel c_1 a un gap plus bas qu'un autre classement consensuel c_2, ça veut dire que c_1 est de meilleure qualité que c_2.

Le problème est que les jeux de données biologiques contiennent généralement trop d'éléments pour calculer un classement consensuel optimal avec un algorithme exact. Dans ce cas, nous utilisons comme référence le classement consensuel avec le score le plus bas (donc celui de meilleure qualité) obtenu par une heuristique. Nous notons ce classement c^+ et utilisons alors le *m-gap* [25] défini ci-dessous.

Définition 3.8.2. *Soit R une liste de classements et c^+ un classement consensuel pris comme référence (possiblement non optimal). Le m-gap est défini de la manière suivante.*

$m\text{-}gap(c, R) = \frac{M(c,R)}{M(c^+,R)} - 1$

Remarque. *S'il se trouve que c^+ est optimal, alors $m\text{-}gap(c, R) = gap(c, R)$.*

Dans cette expérience, nous comparons le $m - gap$ et le temps de calcul des trois heuristiques mentionnées en début de section sur les 1 354 datasets. Pour 60.4 % de nos jeux de données, le $m-$gap est égal au gap puisque nous avons un classement consensuel optimal d'après les expériences précédentes.

La Table 3.6 fournit les résultats du banc d'essai que nous avons effectué. Deux points doivent être remarqués.

- Premièrement, l'utilisation de la phase de partitionnement permet d'augmenter la qualité des algorithmes rapides sans impacter leur temps de calcul de manière significative pour l'utilisateur. Plus précisément, le $m-gap$ (dont la valeur se rapproche de 0 lorsque la qualité du classement consensuel augmente) de KwikSort a été diminué de 47,0 % et le m-gap de CopelandMethod a été diminué de 59 %. Le temps de calcul reste très raisonnable (moins d'une seconde pour KwikSort et CopelandMethod avec le processus de partitionnement).
- Deuxièmement, l'utilisation du partitionnement permet de réduire le temps de calcul des algorithmes de haute qualité, tout en améliorant leur qualité. Plus précisément, le temps de calcul de BioConsert a été réduit de 65 % tandis que son $m - gap$ a été réduit de 30 %.

Heuristique	m-gap moyen	Temps moyen	Temps maximal
$CopelandMethod$	1.14 * 10^{-2}	13 ms	695.5 ms
$ParCons$ avec $CopelandMethod$	6.74 10^{-3}	60 ms	1.4 s
$KwikSort$	1.30 * 10^{-2}	3 ms	19.6 ms
$ParCons$ avec $KwikSort$	6.83 * 10^{-3}	19 ms	983 ms
$BioConsert$	4.61 * 10^{-5}	207 ms	9.7s
$ParCons$ avec $BioConsert$	3.23 * 10^{-5}	72 ms	2.94 s
$ParCons$ avec $exact(SCC < 80)$ et $BioConsert$	2.41 * 10^{-5}	1.0 s	20.5 s $\leq 5s$ dans 93% des datasets

TABLE 3.6 – Banc d'essai : comparaison de la qualité des classements consensuels renvoyés (m-gap) et du temps de calcul. Pour rappel, un classement consensuel de meilleure qualité est associé à un $m-$gap plus proche de 0.

Sur la base de ces expériences, ConQuR-BioV2 exécute systématiquement l'algorithme exact et BioConsert en parallèle. Si l'algorithme exact fournit un classement consensuel en moins de 5 secondes, ce résultat est utilisé. Sinon, c'est le résultat de BioConsert qui est utilisé.

3.8.3 Évaluation de ParFront sur des données biologiques

Les expériences 4 et 5 ont pour objectif d'évaluer la capacité de ParFront à calculer des frontières sur des données biologiques pour lesquelles les frontières sont attendues en début de classement ainsi que de comparer ParFront avec les autres méthodes de l'état de l'art.

Expérience 4 : capacité à calculer des frontières. Les frontières ont été présentées et définies dans la Sous-section 3.6.1. Pour rappel, il s'agit d'un entier positif dont la valeur que l'on note κ indique que l'ensemble des éléments constituant le $top - \kappa$ de tous les classements consensuels optimaux est le même.

Les frontières ont été définies pour aider les utilisateurs à évaluer la confiance qu'ils peuvent avoir vis-à-vis du classement consensuel renvoyé. Les frontières trouvées au début du classement (premières positions) présentent un intérêt particulier car la plupart des utilisateurs se focalisent principalement sur les premiers gènes du classement consensuel.

Trois points ressortent de cette expérience.

- Tout d'abord, 73 % de nos jeux de données ont une frontière en position 1, c'est-à-dire que pour 73 % des maladies, c'est le même gène qui est en première position dans tous les classements consensuels optimaux.
- Ensuite, on peut voir sur la Figure 3.10 qui fournit le nombre moyen de frontières entre 1 et k que nos jeux de données ont en moyenne 10 frontières pour k = 50.
- Enfin, on observe également sur la Figure 3.10 que le nombre moyen de frontières entre les positions 1 et k diminue lorsque k augmente : l'apparition de nouvelles frontières est un évènement qui semble se raréfier à

mesure que l'on avance dans les classements consensuels.

De tels résultats sont particulièrement intéressants car ils démontrent la capacité de ParFront à calculer des frontières utiles à l'utilisateur dans des cas d'utilisation réels.

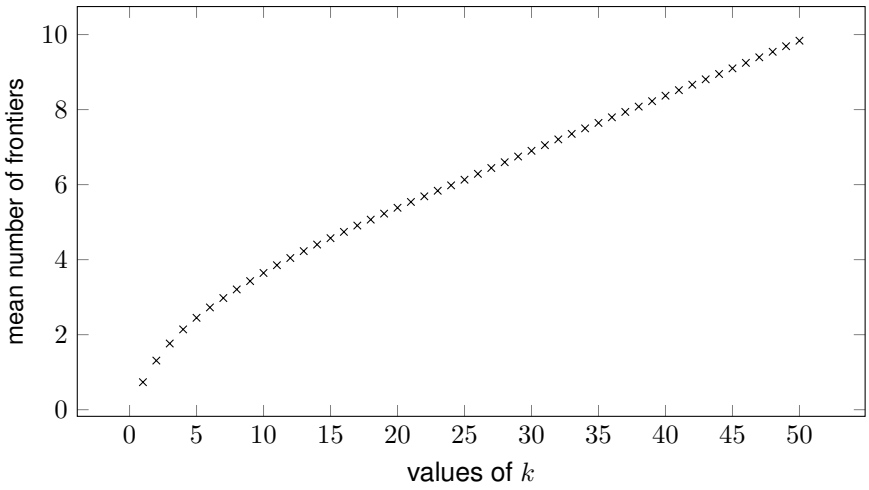


FIGURE 3.10 – Nombre moyen de frontières entre 1 et k, k allant de 1 à 50 (tous les jeux de données ont été considérés).

Expérience 5 : capacité à calculer des frontières que les autres méthodes de l'état de l'art n'arrivent pas à calculer. Dans cette nouvelle expérience dédiée aux frontières, nous considérons l'ensemble des jeux de données du *Dataset 1* et comparons le nombre de frontières obtenues par ParFront avec le nombre de frontières obtenues par les éléments propres (NDE) de Betzler [19], le critère de Condorcet étendu (ECC) [83] et le graphe robuste des éléments (RGE) [7]. Dans un souci d'équité, nous avons considéré les classements unifiés afin de rendre les classements complets : si un gène x est présent dans un classement r alors qu'un autre gène y est manquant dans r, nous avons considéré que x est avant y dans r. Cette considération augmente le nombre de frontières trouvées par les méthodes ECC et NDE avec lesquelles nous nous comparons.

Pour rappel, la propriété 3.6.3 déclare que ParFront trouve nécessairement toutes les frontières données par les méthodes NDE, ECC et RGE (ceci est prouvé dans la Section 3.6.4).

Cette expérience vise à quantifier combien de frontières supplémentaires peuvent être obtenues en utilisant ParFront par rapport aux autres méthodes.

Le résultat, présenté dans la Figure 3.11, est particulièrement intéressant. En effet, ParFront trouve $1,6$ fois plus de frontières que la méthode RGE que vous avons publiée dans [7], presque $15,7$ fois plus de frontières que la méthode ECC (*Extended Condorcet Criterion*, Truchon [83]) et $134,7$ fois plus frontières que la méthode NDE (*Non-Dirty Elements*, Betzler *et al.* [17]).

Autre fait intéressant, la Figure 3.11 montre également que bien que la méthode NDE fournit beaucoup moins de frontières que les méthodes ECC et RGE, NDE est capable de trouver des frontières qui ne sont ni trouvées par

ECC, ni trouvées par RGE. Pour rappel, elles sont nécessairement obtenues par ParFront.

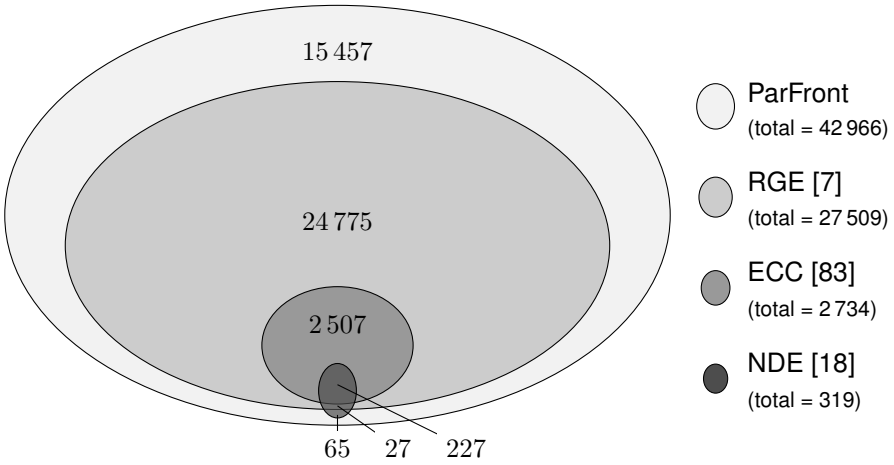


FIGURE 3.11 – Diagramme de Venn indiquant le nombre de frontières obtenues par chacune des quatre méthodes suivantes : NDE (éléments propres, Betzler et al. [18]), critère de Condorcet étendu Truchon [83], RGE (*graphe robuste des éléments*, [7]) et ParFront.

Cette série d'expériences montre la pertinence des algorithmes ParCons et ParFront dans le cadre de l'agrégation de classements incomplets avec égalités. D'une part, grâce au partitionnement, un classement consensuel optimal a été trouvé pour 60 % des jeux de données biologiques massifs pour un temps moyen d'attente de l'ordre de la seconde (par jeu de données). L'utilisation de ParCons couplé à un algorithme exact pour les sous-problèmes de petite taille et l'heuristique BioConsert pour les sous-problèmes de grande taille donne des résultats particulièrement bons du point de vue du score de Kemeny. D'autre part, le Théorème 3.6.1 sur lequel est basé ParFront permet de calculer beaucoup plus de frontières sur les jeux de données biologiques considérés dans ces expériences que les autres méthodes de l'état de l'art.

3.9 Intérêt de l'agrégation de classements pour les données biologiques

Alors que les expériences de la section précédente permettent d'évaluer les différents algorithmes mis en jeu sous le prisme de la méthode Kemeny-Young (score de Kemeny, adapté ici pour gérer les égalités et les classements incomplets), l'objectif de cette section est d'évaluer à quel point l'utilisation des synonymes (et donc l'utilisation de l'agrégation de classements) par le logiciel ConQuR-BioV2 permet de fournir un classement de gènes pertinent vis-à-vis de la maladie d'intérêt. Nous nous intéressons donc ici à la pertinence biologique des classements calculés. Les expériences présentées dans cette section ont été publiées dans [8].

3.9.1 Jeux de données considérés et *gold standards*

Pour les besoin des expériences présentées tout au long de cette section, nous nous sommes servis de la base de données Orphanet qui est une référence mondiale en ce qui concerne les maladies rares.

Présentation d'Orphanet.

Orphanet est un portail spécialisé dans les maladies rares créé en France en 1996. Son objectif est de faciliter le diagnostic et la prise en charge des maladies rares grâce à une base de données publique, permettant ainsi de regrouper de nombreuses informations importantes concernant ces maladies orphelines (gènes associés, médicaments, essais cliniques, centres de prise en charge, ...). L'information contenue sur ce portail de référence est considérée par la communauté biologique, bio-informatique et médicale comme particulièrement fiable en terme de qualité.

La Figure 3.12 montre un extrait de la page correspondant à la maladie rare "Leucémie myéloïde chronique" sur la base de données Orphanet.

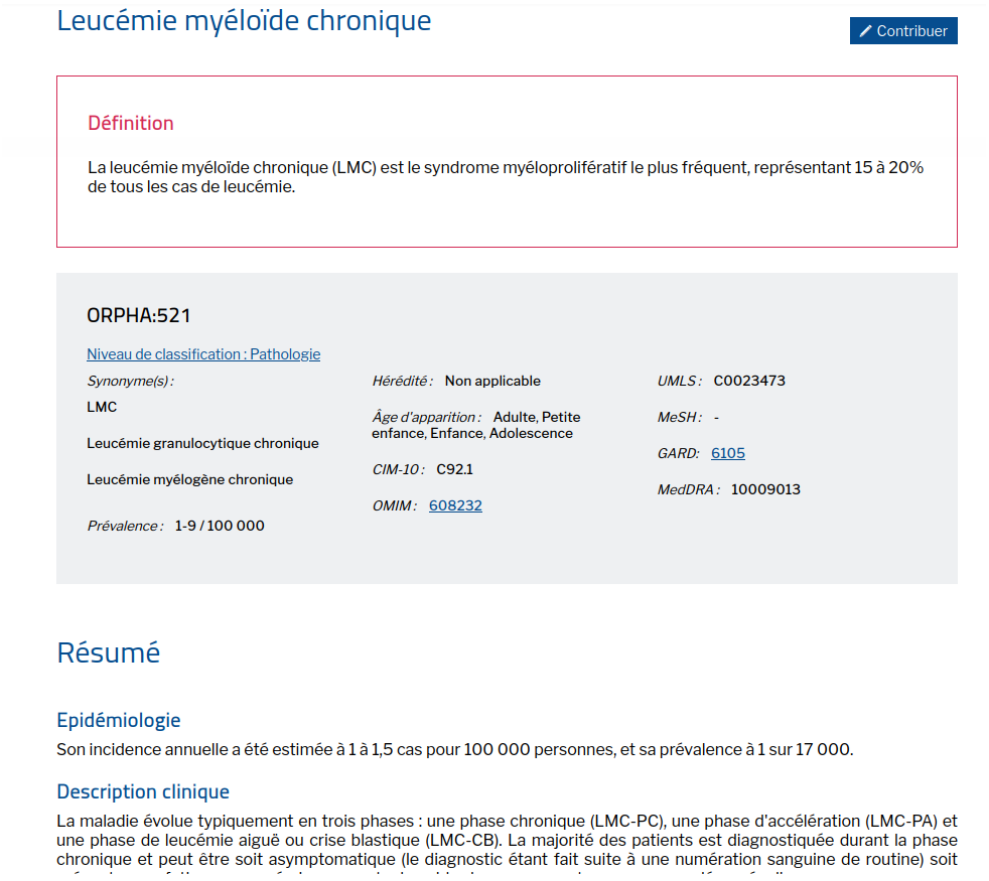


FIGURE 3.12 – Extrait de la page correspondant à la maladie rare "Leucémie myéloïde chronique" sur la base de données Orphanet

Alors que la base de données *Gene* du NCBI peut renvoyer un classement de plusieurs milliers de gènes pour une maladie donnée, la base de données équivalente sur Orphanet ne renvoie que quelques gènes d'intérêt (entre 0

et 30) sans les classer. Les gènes associés à la leucémie myéloïde chronique dans la base de données d'Orphanet sont visibles en Figure 3.13.

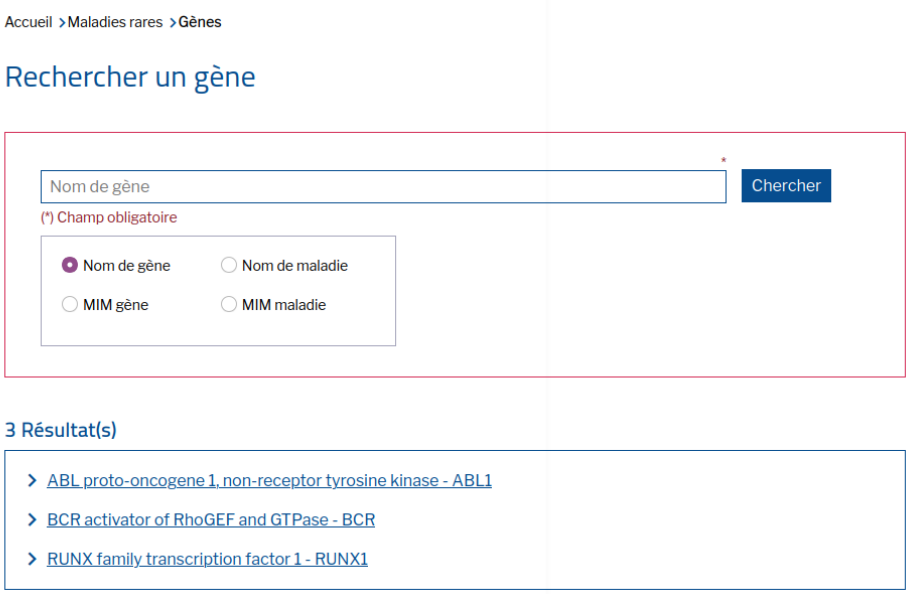


FIGURE 3.13 – Gènes associés à la maladie rare "Leucémie myéloïde chronique" dans la base de données d'Orphanet

Ainsi, cette base peut jouer le rôle de *gold standard* et permettre d'évaluer la qualité des classements consensuels obtenus par nos approches.

Dataset 2 (Reformulations synonymes de maladies avec *gold standard*)**.** *Le Dataset 2 est un ensemble de 234 jeux de données formé d'un sous-ensemble du Dataset 1. Nous avons retenu parmi les 1354 jeux de données du Dataset 1 ceux qui correspondent à des maladies rares et pour lesquels nous disposions d'au moins un gène connu dans Orphanet qui soit également présent dans la base de données Gene du NCBI.*

Il est intéressant de noter qu'Orphanet fournit des informations sur la raison pour laquelle un gène est associé à une maladie. Les informations fournies nous permettent de considérer trois catégories de gènes : (i) les gènes de catégorie 1 qui sont fortement associés à la maladie, ils correspondent à des cibles thérapeutiques ou mutation(s) germinale(s) ou somatique(s) pathogène(s) affectant la fonction du gène, (ii) les gènes de catégorie 2 correspondent aux gènes qui ont un impact sur les manifestations clinique de la maladie ou sont utilisés pour surveiller l'évolution de la maladie, (iii) les gènes de la catégorie 3 sont les gènes "candidats" pour lesquels une mutation est suspectée d'entraîner la maladie ou les gènes jouant un rôle dans les ré-arrangements chromosomiques potentiellement impliqués dans la maladie.

3.9.2 Résultats obtenus par ConQuR-BioV2 par rapport à *Gene* du NCBI

Cette série d'expériences vise à comparer le classement consensuel fourni par la base de données *Gene* du NCBI avec notre approche basée sur l'agrégation de classements qui utilise les reformulations des noms de maladies. Pour ce faire, nous utilisons comme référence la base de données d'Orphanet qui, comme indiqué dans la sous-section 3.9.1, est particulièrement fiable.

Dans l'expérience (a), on se concentre sur les jeux de données. L'objectif est de déterminer lequel de ConQuR-BioV2 et du NCBI fournit le plus de gènes du *gold standard* dans son top-20.

Dans l'expérience (b), on somme le nombre de gènes appartenant au *gold standard* obtenus dans tous les top-20 des classements consensuels de ConQuR-BioV2 et des classements du NCBI.

L'expérience (c) s'intéresse à déterminer laquelle des deux méthodes présente le plus "précocement" dans son top-20 un gène du *gold standard*.

Toutes ces expériences ont également été menées en considérant le top-10, le top-30 et le top-40 plutôt que le top-20 et en considérant uniquement les gènes de catégorie 1, ou les gènes de catégories 1 et 2. Dans tous ces cas, les résultats obtenus étaient très similaires aux résultats présentés. Pour éviter les redondances, ils ne seront pas présentés.

Expérience (a) : Qui de ConQuR-BioV2 ou du NCBI fournit le plus de gènes du *gold standard* ? La Figure 3.14 montre que ConQuR-BioV2 trouve un nombre strictement plus élevé de gènes du *gold standard* dans son top-20 que le NCBI dans 23,1 % des jeux de données tandis que le NCBI trouve un nombre strictement plus élevé de gènes du *gold standard* dans son top-20 que ConQuR-BioV2 dans seulement 3,8 % des jeux de données.

Il est intéressant de remarquer que dans 96,2 % des jeux de données (73,08 % + 23,08 %, correspondant à 225 jeux de données), ConQuR-BioV2 trouve le même nombre ou un nombre strictement supérieur de gènes du *gold standard* dans son top-20 que le NCBI.

Expérience (b) : Nombre total de gènes du *gold standard*. La Figure 3.15 fournit le nombre total de gènes du *gold standard* (additionné sur tous les jeux de données) obtenus dans le top-20 de ConQuR-BioV2 et du NCBI : ConQuR-BioV2 contient 1,27 fois plus de gènes du *gold standard* que le NCBI. De plus, le nombre de gènes trouvés par ConQuR-BioV2 uniquement est deux fois plus important que le nombre de gènes trouvés par le NCBI seulement. Enfin (non représenté sur la figure), on peut ajouter que le NCBI ne trouve aucun gène du *gold standard* dans 44 jeux de données alors que ce n'est le cas que dans 12 jeux de données pour ConQuR-BioV2.

Expérience (c) : Rang du gène du *gold standard* le mieux positionné. La dernière expérience se focalise sur la position du gène le mieux classé du *gold standard* pour chacune des deux méthodes. La Figure 3.16 montre que

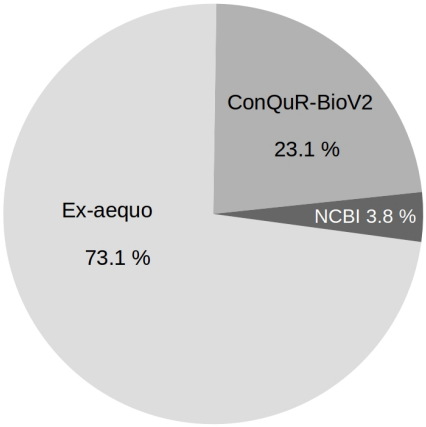


FIGURE 3.14 – Nombre de jeux de données tels que (i) le top-20 de ConQuR-BioV2 contient strictement plus de gènes du *gold standard* que le top-20 du NCBI, (ii) le top-20 du NCBI contient strictement plus de gènes du *gold standard* que le top-20 de ConQuR-BioV2,(iii) ConQuR-BioV2 et le NCBI contiennent autant de gènes du *gold standard* dans leur top-20 (Ex-aequo).

le gène le mieux classé du *gold standard* retourné par ConQuR-BioV2 est classé strictement mieux que le gène le mieux classé du *gold standard* retourné par le NCBI dans 26,5 % des jeux de données.

Il est intéressant de noter grâce à la Figure 3.16 que le gène le mieux classé de ConQuR-BioV2 appartenant au *gold standard* est classé de façon similaire ou mieux classé que le gène le mieux classé de NCBI appartenant au *gold standard* dans 91 % des jeux de données.

Ces expériences mettent en évidence que l'agrégation de classements rendue possible par l'utilisation des reformulations synonymes des maladies permet d'améliorer significativement le classement renvoyé par rapport au classement du NCBI ne se servant pas des reformulations synonymes.

3.10 Conclusion

Extraire de l'information à partir des requêtes sur les dénominations synonymes d'une même maladie aboutit au problème d'agréger des classements qui sont non seulement incomplets (des gènes sont associés à certains synonymes et pas à d'autres) mais également avec égalités (deux gènes peuvent être aussi pertinents l'un que l'autre vis-à-vis d'un synonyme donné et donc positionnés au même rang). Nous avons abordé ce problème en utilisant le score de Kemeny généralisé à la pseudo-distance défini pour les besoins de ce cas d'utilisation dans [24]. Nos contributions sont rappelées ci-dessous.

Nous avons tout d'abord conçu et implémenté l'algorithme ParCons capable de diviser le problème de départ en sous-problèmes. Il est alors possible d'appeler sur ces sous-problèmes un algorithme exact s'il y a moins de 80 éléments, et une heuristique sinon. Cela a été rendu possible par notre représentation des classements d'entrée par un graphe dit *graphe des éléments*, graphe dont on a démontré que les composantes fortement connexes triées

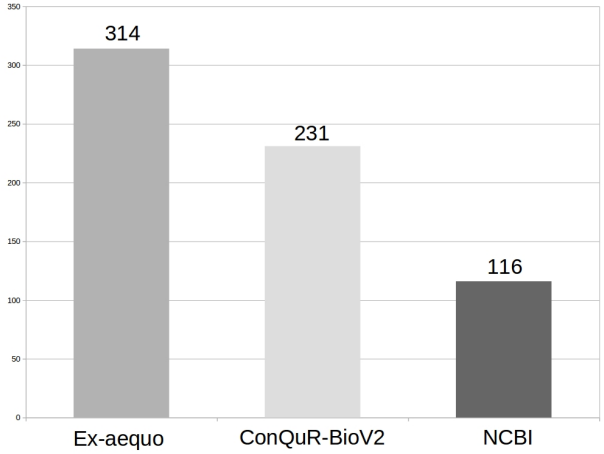


FIGURE 3.15 – Somme sur tous les jeux de données du nombre de gènes du *gold standard* (i) dans le top-20 de ConQuR-BioV2 et du NCBI, (ii) dans le top-20 de ConQuR-BioV2 uniquement, (iii) dans le top-20 du NCBI uniquement.

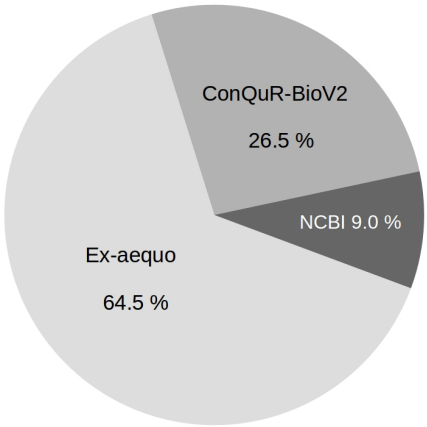


FIGURE 3.16 – Nombre de jeux de données tels que la position du premier gène appartenant au *gold standard* dans le top-20 du classement consensuel est (i) meilleure que (ConQur-BioV2), (ii) moins bien que ConQuR-BioV2, ou (iii) similaire (Ex-aequo) à la position du premier gène du NCBI appartenant au *gold standard*.

selon un ordre topologique représentent des sous-problèmes indépendants.

Le second algorithme que nous proposons, l'algorithme ParFront, répond au problème de la non-unicité d'un classement consensuel optimal en fournissant à l'utilisateur un indice quant au niveau de confiance qu'il peut avoir envers le classement consensuel calculé. Nous avons pour cela défini le concept de frontières : si l'entier k est une frontière, alors l'ensemble des k premiers éléments est identique pour tous les classements consensuels optimaux. Notre approche inclut toutes les frontières obtenues par les éléments propres de Betzler *et al.* [17] et le critère de Condorcet étendu [83], et nous avons observé expérimentalement que le nombre de frontières que nous calculons est bien supérieur à celui obtenu par ces deux dernières méthodes.

La pertinence de notre approche a été montrée en l'appliquant sur plus d'un millier de jeux de données biolo-

giques de grande taille. Un classement consensuel optimal a été trouvé pour plus de 60 % des jeux de données. Par ailleurs, sur les jeux de données correspondant à des maladies rares pour lesquels un *gold standard* est fourni par Orphanet, notre méthode a fréquemment mis en avant des gènes d'intérêt qui étaient mal classés sans la prise en compte des synonymes par l'agrégation de classements.

Les algorithmes ParCons et ParFront ont été implémentés dans l'outil en ligne ConQuR-BioV2 disponible à l'adresse suivante : `http://conqur-bio.lri.fr/`.

De nombreuses perspectives sont possibles pour faire suite à ces travaux, nous en dégageons deux ici.

Une première perspective est de compléter l'approche des frontières par d'autres indicateurs de confiance concernant la flexibilité de la position des éléments dans les classements consensuels optimaux. Deux approches complémentaires sont envisageables. Une première serait d'exploiter les différents tris topologiques du graphe des composantes fortement connexes issu du graphe des éléments. Pour rappel, à chaque tri topologique correspond au moins un classement consensuel optimal distinct. Les tris topologiques sont donc susceptibles de mettre en avant des éléments dont la position dans les classements consensuels optimaux varie particulièrement, auquel cas une mise en garde pourrait être donnée à l'utilisateur. Une deuxième approche serait de généraliser la méthode à base de bootstrap présentée dans [51] sur les classements incomplets avec égalités afin de calculer des intervalles de confiance sur la position des éléments dans le classement consensuel. L'intérêt de cette dernière méthode est sa capacité à évaluer à quel point de petits changements dans les classements de départ sont susceptibles d'induire des changements importants dans les classements consensuels optimaux.

La seconde perspective porte sur l'évaluation de la pertinence des classements consensuels renvoyés par notre approche. Nous souhaitons d'une part étudier de façon plus approfondie les caractéristiques des jeux de données sur lesquels nos approches sont efficaces, en reproduisant les expériences menées dans ce chapitre sur des données biologiques de natures différentes ou sur des jeux de données réels non biologiques. D'autre part, une perspective pluridisciplinaire importante vise à évaluer nos résultats en fonction de l'adéquation des gènes renvoyés dans les premières positions par ConQur-BioV2 et les annotations associées à ces gènes. Dans le cas où aucun *gold standard* (comme Orphanet) n'est disponible, cela peut offrir un moyen de quantifier la pertinence des résultats.

Résumé du Chapitre 3

Les classements issus de données biologiques que l'on considère dans ce chapitre sont incomplets (certains éléments sont présents dans certains classements et absents d'autres) et avec égalités (certains éléments peuvent être *ex-aequo* dans certains classements). Nous avons utilisé le score de Kemeny généralisé à la pseudo-distance défini dans dans [24] pour les besoins du cas d'utilisation considéré ici.

Nous avons établi, grâce à une représentation des classements de départ par un graphe, une propriété suffisante permettant de partitionner le problème de départ en sous-problèmes indépendants : la concaténation d'un classement consensuel optimal de chaque sous-problème forme un classement consensuel optimal pour le problème de départ. Nous avons alors conçu l'algorithme ParCons dont la première étape consiste à diviser le problème de départ en sous-problèmes et dont la deuxième étape consiste à appeler un algorithme auxiliaire pour calculer un classement consensuel pour les sous-problèmes non triviaux (de taille supérieure ou égale à 3). Lorsque la taille du sous-problème est petite (en pratique, inférieure à 80), un algorithme exact est utilisé. Pour les sous-problèmes de grande taille, il est nécessaire d'appeler une heuristique.

Nous avons également défini le concept de frontières de la manière suivante : un entier k est une frontière si et seulement si l'ensemble des éléments du top-k est le même dans tous les classements consensuels optimaux. Une frontière est donc un indicateur de robustesse : elle garantit que tous les éléments du top-k sont devant tous les autres éléments dans tous les classements consensuels optimaux. En considérant un sous-ensemble des arcs du graphe des éléments appelés arcs robustes, nous avons conçu et implémenté l'algorithme ParFront permettant de calculer plus de frontières que n'importe quelle autre méthode de l'état de l'art. Nous avons démontré en particulier que ParFront retrouve toutes les frontières obtenues par les éléments propres de Betzler *et al.* [17] et le critère de Condorcet étendu [83], et observé expérimentalement que le nombre de frontières obtenues par notre approche est bien supérieur à celles obtenues par ces deux dernières méthodes sur nos jeux de données biologiques.

ParCons et ParFront ont été évalués sur 1354 jeux de donnés réels de grande taille (entre 100 et 1557 éléments) correspondant au contexte biologique considéré. ParCons a permis qu'un classement consensuel optimal soit trouvé pour 60 % des jeux de données.

Enfin, nous avons montré l'intérêt d'utiliser l'agrégation de classements pour ce cas d'utilisation biologique en considérant un *gold standard*. En prenant comme référence la base de données Orphanet dont les informations sont considérées comme très fiables, nous avons pu conclure que les top-20 des classements consensuels renvoyés par ParCons contiennent plus de gènes connus pour être fortement liés aux maladies considérées que les top-20 des classements obtenus par la base de données *Gene* du NCBI qui n'utilise pas les synonymes (et le premier gène du *gold standard* qui apparaît dans le classement consensuel renvoyé par ConQuR-BioV2 est mieux positionné que le premier gène du *gold standard* qui apparaît dans le classement consensuel renvoyé par *Gene*).

Les algorithmes ParCons et ParFront ont été implémentés dans l'outil en ligne ConQuR-BioV2 disponible à l'adresse suivante : `http://conqur-bio.lri.fr/`.

Chapitre 4

Modèle général pour l'agrégation de classements incomplets

Sommaire

4.1 Introduction

Dans le chapitre précédent, nous nous sommes intéressés à un contexte biologique précis : l'utilisation de reformulations synonymes d'une même maladie dans le but de déterminer les gènes les plus pertinents vis-à-vis de cette maladie. Les techniques d'agrégation de classements ont été utilisées dans ce contexte pour la première fois dans [24]. Dans cet article, pour pallier le problème des données manquantes, Brancotte *et al.* ont défini et utilisé la pseudo-distance de Kendall-τ généralisée (voir Définition 2.3.5) qui est une généralisation de la distance de Kendall-τ aux classements complets avec égalités. Or, plusieurs autres méthodes ont été proposées dans la littérature pour l'agrégation de classements incomplets [4, 44, 78].

4.1.1 Plusieurs méthodes pour gérer les données manquantes

Différentes alternatives pour gérer les classements incomplets existent dans la littérature et peuvent être divisées en deux groupes : celles qui consistent à rendre artificiellement complets des classements qui ne l'étaient pas initialement [19, 78] et celles qui consistent à généraliser la distance de Kendall-τ et/ou le score de Kemeny afin de les rendre compatibles avec les classements incomplets. La mesure de Kemeny induite [44] (Définition 2.3.3), la distance de Kendall-τ étendue [4] (Définition 2.3.7) et la pseudo-distance de Kendall-τ généralisée appartiennent ainsi au deuxième groupe de méthodes. Un utilisateur devant agréger des classements incomplets peut légitimement se demander quelle méthode choisir parmi celles-ci. Sont-elles équivalentes en terme de qualité ou certaines méthodes valent-elles mieux que d'autres ?

4.1.2 Comparaison des méthodes

Les différentes méthodes pour gérer les classements incomplets sont parfois très différentes les unes des autres. Pour l'illustrer, comparons le score de Kemeny étendu [4] (Définition 2.3.8) avec le score de Kemeny généralisé à la pseudo-distance [24] (Définition 2.3.6). Considérons un classement d'entrée tel qu'un élément x y est présent et un élément y en est absent.

Pour la première méthode, seules sont considérées les paires (x, y) telles que x et y sont tous les deux présents dans les deux classements à comparer. Ainsi, le coût de mettre x avant y, x après y, x à égalité avec y dans le classement consensuel sera nul. Pour la deuxième méthode, en revanche, le fait de mettre y avant x dans le consensus va entraîner un coût de 1.

Cette illustration met en évidence que ces deux adaptations du score de Kemeny sont différentes. On peut supposer que le choix de l'une ou l'autre de ces adaptations peut avoir un impact important sur le classement consensuel qui sera calculé.

4.1.3 Besoin d'apporter un regard qualitatif sur les données

Finalement, la question qui se pose est la suivante : comment doivent être considérés les éléments manquants vis-à-vis des éléments présents ? Selon le contexte, la réponse peut être différente.

Prenons le cas d'un étudiant inscrit à l'université qui n'a pas choisi une option. Il n'y a pas lieu de le comparer vis-à-vis de cette option avec les étudiants qui ont suivi l'option. Dans un tel cas, les éléments présents sont incomparables avec les éléments absents. En particulier, les étudiants non inscrits à une option ne doivent pas être considérés comme "derniers" de l'option. A l'inverse, dans le contexte du chapitre précédent, lorsqu'un gène est absent d'un classement, on peut conclure que la base de données l'a jugé non pertinent vis-à-vis de la requête. On peut donc légitimement le considérer comme moins pertinent que les gènes présents.

Nous allons montrer dans ce chapitre que c'est un regard qualitatif sur les données qui permet de choisir quelle adaptation du score de Kemeny est la plus adaptée à un contexte donné.

4.1.4 Besoin d'un cadre algorithmique pour les classements incomplets

Les généralisations du score de Kemeny aux classements incomplets avec égalités ne sont pas associées à un algorithme exact. De plus, les méthodes de partitionnement et plus généralement les techniques de réduction d'espace présentées au Chapitre 2 ne s'appliquent pas sur les classements incomplets. En définitive, seules des heuristiques sont utilisées dans le cadre de l'agrégation de classements incomplets avec égalités. Or, un utilisateur a besoin de savoir à quel point le classement consensuel est de bonne qualité et robuste.

4.1.5 Contributions

Pour répondre aux besoins explicités précédemment, nous présentons un modèle paramétré incluant les adaptations du score de Kemeny citées précédemment et permettant d'apporter un regard qualitatif sur les données. Ce regard qualitatif permet de choisir des paramètres adaptés à un contexte donné, et ainsi de considérer une généralisation du score de Kemeny pertinente vis-à-vis de ce contexte. Nous avons également conçu un algorithme exact générique et généralisé plusieurs heuristiques pour le cadre de l'agrégation de classements incomplets. Les algorithmes ParCons et ParFront présentés au chapitre précédent ont en particulier été adaptés pour être utilisés au sein de ce modèle. Nous avons étudié axiomatiquement notre modèle en nous basant sur les critères d'équité de la théorie du choix social. Ce cadre générique a été implémenté dans un outil en ligne appelé CoRankCo. Plusieurs expériences ont été menées sur des jeux de données de natures différentes afin d'évaluer le modèle. Les travaux présentés dans ce chapitre font l'objet d'un article en cours de rédaction.

Ce chapitre est organisé de la manière suivante. La Section 4.2 présente notre modèle pour gérer les classements incomplets incluant les méthodes citées précédemment et permettant d'en concevoir de nouvelles. Les paramètres de ce modèle permettent de gérer des jeux de données issus de contextes très différents (données biologiques, bulletins de vote, classements de films, ...). La Section 4.3 propose des algorithmes utilisables pour ce modèle, dont un algorithme exact et plusieurs heuristiques. La Section 4.4 étudie le modèle d'un point de vue axiomatique en énonçant les conditions nécessaires et suffisantes sur les paramètres du modèle pour respecter certains axiomes importants de la communauté théorie du choix social. La Section 4.5 présente CoRankCo, l'outil en ligne dans lequel le modèle présenté en Section 4.2 et les algorithmes présentés en Section 4.3 ont été implémentés. Enfin, la Section 4.6 évalue le modèle à travers plusieurs expériences sur des jeux de données de différentes natures.

Notation. *Dans ce chapitre, nous utiliserons régulièrement la notation suivante très utilisée en théorie du choix social : pour un ensemble X, $X^{<\infty}$ correspondra à $\bigcup_{i\geq 1} X^i$.*

4.2 Modèle pour l'agrégation de classements incomplets

L'objectif de cette section est de présenter un nouveau modèle pour l'agrégation de classements incomplets incluant la méthode d'unification ainsi que les différentes généralisations du score de Kemeny présentées dans la Section 2.3. Le modèle proposé ci-dessous permet également de concevoir d'autres généralisations du score de Kemeny aux classements incomplets avec égalités. L'intérêt d'un tel modèle est entre autres de permettre la conception d'algorithmes utilisables quelque soit la généralisation choisie.

4.2.1 Définition du modèle

Nous détaillons ici les paramètres du modèle correspondant aux différentes pénalités qui vont être sommées afin d'obtenir le score de Kemeny adapté aux classements incomplets. Ce modèle comporte plusieurs paramètres. Deux d'entre eux correspondent au paramètre p du modèle de Fagin *et al.* [47] et correspondent au coût de créer ou de rompre une égalité (ces coûts peuvent être différents dans le modèle que nous présentons contrairement au modèle de Fagin).

Dans le modèle de Fagin, pour chaque paire d'éléments (x, y) du classement consensuel telle que x et y sont à égalité, il faut payer p pour chaque classement d'entrée tel que x et y ne sont pas à égalité (rupture d'égalité). De la même manière, pour chaque paire (x, y) d'éléments tels que x et y ne sont pas à égalité dans le classement consensuel, il faut payer p pour chaque classement d'entrée tel que x et y sont à égalité (création d'égalité).

Ce modèle est une référence pour la gestion des égalités, mais n'est pas utilisable lorsque les classements d'entrée sont incomplets. Or, gérer proprement les classements incomplets est particulièrement complexe. En effet, le nombre de cas à traiter (donc de pénalités possibles) est beaucoup plus important que dans le cas des classements complets avec égalités. Les différents cas possibles sont détaillés ci-dessous, et les pénalités associées correspondent aux paramètres de notre modèle.

Même si les classements d'entrée sont incomplets, le classement consensuel noté ici c doit être complet par définition. En conséquence, pour toute paire (x, y) d'éléments de U, on a 3 possibilités :

- x est avant y dans c, noté $x \prec_c y$.
- y est avant x dans c, noté $y \prec_c x$.
- x et y sont à égalité dans c, noté $x \equiv_c y$.

Contrairement au classement consensuel, un classement d'entrée r peut être incomplet. En conséquence, pour toute paire (x, y) d'éléments de U, on a cette fois 6 possibilités :

- x est avant y dans r, noté $x \prec_r y$.
- y est avant x dans r, noté $y \prec_r x$.
- x et y sont à égalité dans r, noté $x \equiv_r y$.
- x est présent et y est absent, noté $x \diamond_r y$.
- y est présent et x est absent, noté $y \diamond_r x$.
- x et y sont tous les deux absents.

L'idée du modèle est la suivante : pour une paire (x, y) d'éléments de U, on va associer une pénalité pour chaque ordre relatif possible dans le classement consensuel c et chaque ordre relatif possible dans le classement d'entrée r. Notons que $x \prec_c y$ et $y \prec_c x$ peuvent être rassemblés en un seul et même cas car ils sont symétriques.

Ainsi, nous définissons deux vecteurs B et T tels que $B \in \mathbb{R}^6$ et $T \in \mathbb{R}^6$. Le vecteur B (pour *before*) va contenir les pénalités associées aux paires (x, y) telles que x est avant y dans le classement consensuel. Le vecteur T

(pour *tied*) va contenir les pénalités associées aux paires (x, y) telles que x et y sont à égalité dans le classement consensuel. Les pénalités sont explicitées dans la Table 4.1. Nous pouvons par exemple voir dans cette Table que si un élément x est avant un autre élément y dans le classement consensuel alors que dans un classement d'entrée r on a $x \diamond_r y$ (x présent dans r alors que y est absent), la pénalité à payer est $B[4]$ c'est à dire la valeur correspondant à la quatrième composante du vecteur B.

TABLE 4.1 – Table des pénalités du modèle

$r \in R$ \ $c \in C(U)$	$x \prec_c y$	$x \equiv_c y$
$x \prec_r y$	$B[1]$	$T[1]$
$y \prec_r x$	$B[2]$	$T[2]$
$x \equiv_r y$	$B[3]$	$T[3]$
$x \diamond_r y$	$B[4]$	$T[4]$
$y \diamond_r x$	$B[5]$	$T[5]$
$x \notin dom(r) \wedge y \notin dom(r)$	$B[6]$	$T[6]$

Nous définissons maintenant le *Score de Kemeny $S^{(B,T)}$ avec vecteurs de pénalité B et T*, permettant de mesurer à quel point un classement consensuel c représente bien une liste de classements R au vu des pénalités choisies.

Définition 4.2.1 (Score de Kemeny $S^{(B,T)}$ avec vecteurs de pénalité B et T)**.** *Nous définissons le score de Kemeny $S^{(B,T)}$ avec vecteurs de pénalité B et T de la manière suivante : pour tout $(c, R) \in C(U) \times A(U)^{<\infty}$,*

$$S^{(B,T)}(c, R) = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \langle B, \Omega^R_{x,y} \rangle \; + \; \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} \langle T, \Omega^R_{x,y} \rangle \tag{4.1}$$

avec $\Omega^R_{x,y}$ le vecteur $(\omega^R_{x \prec y}, \omega^R_{y \prec x}, \omega^R_{x \equiv y}, \omega^R_{x.\overline{y}}, \omega^R_{y.\overline{x}}, \omega^R_{\emptyset})$, avec

- $\omega^R_{x \prec y} = |\{r \in R : x \prec_r y\}|$ *soit le nombre de classements de R tels que x est avant y,*
- $\omega^R_{x \equiv y} = |\{r \in R : x \equiv_r y\}|$ *soit le nombre de classements de R tels que x et y sont à égalité,*
- $\omega^R_{x.\overline{y}} = |\{r \in R : x \diamond_r y\}|$ *soit le nombre de classements de R tels que x est présent alors que y ne l'est pas,*
- $\omega^R_{\emptyset} = |\{r \in R : x \notin dom(r) \wedge y \notin dom(r)\}|$ *soit le nombre de classements de R tels que ni x ni y ne sont présents.*

où $\langle B, \Omega^R_{x,y} \rangle$ (respectivement $\langle T, \Omega^R_{x,y} \rangle$) correspond au produit scalaire de B (respectivement T) avec $\Omega^R_{x,y}$.

Intuitivement, la valeur de $\langle B, \Omega^R_{x,y} \rangle$ représente le coût de placer x avant y dans le classement consensuel c vis-à-vis de R, et la valeur de $\langle T, \Omega^R_{x,y} \rangle$ représente le coût de mettre à égalité x et y dans le classement consensuel c vis-à-vis de R.

Le choix du coefficient 1/2 est motivé par le fait que pour deux éléments distincts x et y tels que $x \equiv_c y$, on a aussi $y \equiv_c x$. Ce coefficient 1/2 nous assure ainsi que la situation $x \equiv_c y$ ne soit bien comptée qu'une seule fois.

Notation. *Afin d'améliorer la lisibilité des formules et calculs, nous utiliserons les notations suivantes. Pour une paire* (x, y) *d'éléments distincts de* U*, nous définissons :*

- $before(x, y) = \langle B, \Omega^R_{x,y} \rangle$ *le coût de placer* x *avant* y *dans le classement consensuel* c *vis-à-vis de* R*.*
- $tied(x, y) = \langle T, \Omega^R_{x,y} \rangle$ *le coût de mettre à égalité* x *et* y *dans le classement consensuel* c *vis-à-vis de* R*.*
- $min(x, y) = min(before(x, y), before(y, x), tied(x, y))$ *le coût du placement relatif le moins cher pour la paire* (x, y)*.*

Un algorithme implémenté en python permettant de calculer $S^{(B,T)}$ de façon efficace est donné en annexe. Cet algorithme est une généralisation de celui inspiré du tri fusion permettant de calculer le score de Kemeny avec une complexité $O(|R| * |U| * \log |U|)$.

Nous définissons maintenant la notion de fonction Kemeny compatible. L'idée principale est d'ajouter des restrictions sur les valeurs des coefficients afin d'éviter des situations non souhaitables.

Définition 4.2.2 (fonction Kemeny compatible)**.** *On dit qu'une fonction* $f : dom(f) \to \mathbb{R}_+$ *est une fonction Kemeny compatible (KCF) si* $P(U) \times P(U)^{<\infty} \subseteq dom(f) \subseteq C(U) \times A(U)^{<\infty}$ *et s'il existe* $B, T \in \mathbb{R}^6_{\geq 0}$ *tels que*

- *(i)* $B[1] = T[3] = 0$*,*
- *(ii)* $B[2] > 0$*,*
- *(iii)* $T[1] = T[2]$*,*
- *(iv)* $T[4] = T[5]$*,*

et tels que $f(c, R) = S^{(B,T)}(c, R)$ *pour tout* $(c, R) \in dom(f)$*.*

Une KCF est donc une fonction qui prend en entrée un classement consensuel (classement complet de U) ainsi qu'une liste de classements de U et renvoie un réel correspondant au score du classement consensuel vis-à-vis de la liste de classements. Plus ce score est élevé, moins le classement consensuel est représentatif des classements d'entrée. Une KCF n'a pas besoin d'être définie sur les classements incomplets ou les classements avec égalité mais se doit malgré tout d'être définie au moins sur l'ensemble des jeux de données correspondant à des permutations.

On peut voir qu'une KCF respecte un certain nombre de contraintes. La restriction (i) garantit qu'aucune pénalité n'est appliquée pour une paire (x, y) si leur ordre relatif est le même entre le classement consensuel c et un classement d'entrée r. Au contraire, la restriction (ii) garantit qu'une pénalité est appliquée si $x \prec_c y$ tandis que $y \prec_r x$. Intuitivement, en prenant $B[2] = 1$, on retombe sur le score de Kemeny classique si les classements d'entrée sont complets et sans égalités (les autres coefficients n'ont alors pas d'incidence). Enfin, les restrictions (iii) et (iv) sont deux conditions nécessaires et suffisantes pour garantir que $tied(x, y) = tied(y, x)$.

Il est intéressant de noter que de manière analogue (i) au score de Kemeny qui somme la distance de Kendall-τ entre le classement consensuel et chaque classement d'entrée, (ii) au score de Kemeny généralisé qui somme la distance de Kendall-τ généralisée avec pénalité p entre le classement consensuel et chaque classement d'entrée,

(iii) au score de Kemeny induit qui somme la distance de Kendall-τ induite entre le classement consensuel et chaque classement d'entrée (et ainsi de suite), le calcul de $S^{(B,T)}(c,R)$ peut se décomposer en une somme sur chaque classement r de R.

Notation. *Dans la suite de ce chapitre, pour un classement r et deux éléments x et y de U, nous notons $\mathbf{1}_{x \prec_r y}$ la fonction qui renvoie 1 si $x \prec_r y$ et 0 dans le cas contraire. Plus généralement, nous notons $\mathbf{1}_a$ la fonction qui renvoie 1 si a est vrai, et 0 dans le cas contraire.*

Propriété 4.2.1. *Soit $f = S^{(B,T)}(c,R)$ une fonction Kemeny compatible. Alors, il existe une fonction g telle que pour tout $(c,R) \in dom(f)$, on a*

$$f(c,R) = \sum_{r \in R} g(c,r). \tag{4.2}$$

Démonstration : D'après la Définition 4.2.2, on sait que $\forall (c,R) \in dom(f)$,

$$\begin{aligned} f(c,R) &= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \langle B, \Omega^R_{x,y} \rangle \; + \; \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} \langle T, \Omega^R_{x,y} \rangle \\ &= \sum_{x \neq y \in U} \mathbf{1}_{x \prec_c y} * \langle B, \Omega^R_{x,y} \rangle \; + \frac{1}{2} * \mathbf{1}_{x \equiv_c y} * \langle T, \Omega^R_{x,y} \rangle \end{aligned}$$

Par définition de B, T et $\Omega^R_{x,y}$ (voir Table 4.1, Définition 4.2.1 et Définition 4.2.2), on obtient :

$$\begin{aligned} f(c,R) &= \sum_{r \in R} \sum_{x \neq y \in U} \mathbf{1}_{x \prec_c y} * (\mathbf{1}_{y \prec_r x} * B[2] + \mathbf{1}_{x \equiv_r y} * B[3] + \mathbf{1}_{x \diamond_r y} * B[4] + \mathbf{1}_{y \diamond_r x} * B[5] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * B[6]) \\ &\qquad + \frac{1}{2} * \mathbf{1}_{x \equiv_c y} * (\mathbf{1}_{x \prec_r y \vee y \prec_r x} * T[1] + \mathbf{1}_{x \diamond_r y \vee y \diamond_r x} * T[4] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * T[6]) \\ &= \sum_{x \neq y \in U} \sum_{r \in R} \mathbf{1}_{x \prec_c y} * (\mathbf{1}_{y \prec_r x} * B[2] + \mathbf{1}_{x \equiv_r y} * B[3] + \mathbf{1}_{x \diamond_r y} * B[4] + \mathbf{1}_{y \diamond_r x} * B[5] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * B[6]) \\ &\qquad + \frac{1}{2} * \mathbf{1}_{x \equiv_c y} * (\mathbf{1}_{x \prec_r y \vee y \prec_r x} * T[1] + \mathbf{1}_{x \diamond_r y \vee y \diamond_r x} * T[4] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * T[6]) \\ &= \sum_{r \in R} g(c,r) \end{aligned}$$

avec

$$\begin{aligned} g(c,R) = \sum_{x \neq y \in U} & \mathbf{1}_{x \prec_c y} * (\mathbf{1}_{y \prec_r x} * B[2] + \mathbf{1}_{x \equiv_r y} * B[3] + \mathbf{1}_{x \diamond_r y} * B[4] + \mathbf{1}_{y \diamond_r x} * B[5] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * B[6]) \\ & + \frac{1}{2} * \mathbf{1}_{x \equiv_c y} * (\mathbf{1}_{x \prec_r y \vee y \prec_r x} * T[1] + \mathbf{1}_{x \diamond_r y \vee y \diamond_r x} * T[4] + \mathbf{1}_{x \notin dom(r) \wedge y \notin dom(r)} * T[6]) \end{aligned}$$

Cette fonction g sera appelée *fonction auxiliaire de f*.

Nous définissons maintenant la notion de classement consensuel optimal dans le cadre du modèle proposé :

Définition 4.2.3 (classement consensuel optimal)**.** *Soit R une liste de classements (possiblement incomplets) et soit f une KCF. Un classement consensuel optimal de R par rapport à f est un classement complet c tel que $(c, R) \in dom(f)$ et $f(c, R) \leq f(r, R)$ pour chaque classement complet r tel que $(r, R) \in dom(f)$.*

En d'autres termes, un classement consensuel optimal de R par rapport à une KCF f est un classement complet tel qu'aucun autre classement complet ne peut mieux représenter R vis-à-vis de f. Comme pour les permutations et des classements complets avec égalités, un classement consensuel optimal n'est pas nécessairement unique.

Nous voulons par ailleurs gérer le cas où un utilisateur exigerait que le classement consensuel soit sans égalités en plus d'être complet (c'est souvent le cas dans un contexte d'élection où on souhaite qu'il y ait un unique vainqueur). Nous définissons donc les ℓ-consensus optimaux :

Définition 4.2.4 (ℓ-classement consensuel optimal)**.** *Soit R une liste de classements (possiblement incomplets) et soit f une KCF. Un ℓ-classement consensuel optimal de R par rapport à f est un classement $\pi \in P(U)$ tel que $(\pi, R) \in dom(f)$ et $f(\pi, R) \leq f(\sigma, R)$ pour chaque $\sigma \in P(U)$ tel que $(\sigma, R) \in dom(f)$.*

En d'autres termes, un ℓ-classement consensuel optimal est une permutation de U tel qu'aucune autre permutation de U ne peut mieux représenter R vis-à-vis de f. Il peut y avoir plusieurs ℓ-classements consensuels optimaux pour une même liste de classements d'entrée.

Notation. *Dans les sections suivantes, nous notons respectivement M_f et Λ_f l'ensemble de tous les classements consensuel optimaux et ℓ-consensus optimaux de R vis-à-vis de f.*

4.2.2 Généricité du modèle

Nous montrons maintenant que ce cadre englobe le score de Kemeny défini sur des permutations (Définition 1.2.2), la généralisation de Fagin *et al.* pour gérer les égalités (Définition 1.3.2), la technique d'unification pour rendre artificiellement complets des classements qui ne le sont pas (Définition 2.3.2) mais également les généralisations du score de Kemeny pour les classements incomplets présentées dans la Section 2.3 du Chapitre 2 (Définitions 2.3.4, 2.3.6, 2.3.8).

Propriété 4.2.2. *Le score de Kemeny (Définition 1.2.2) et le score de Kemeny avec pénalité p (Définition 1.3.2) sont des KCFs.*

Démonstration : Dans la mesure où le score de Kemeny avec pénalité p est une généralisation du score de Kemeny, nous montrons simplement que S^p le score de Kemeny avec pénalité p dont le domaine de définition est $C(U) \times C(U)^{<\infty}$ est une KCF.

Soient $B = (0, 1, p, 0, 0, 0)$ et $T = (p, p, 0, 0, 0, 0)$. Soit $\mathscr{P}$ l'ensemble de toutes les paires non ordonnées de U. Soit $(c, R) \in dom(S^p)$. Nous savons que :

$$
\begin{aligned}
S^{(B,T)}(c,R) &= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} before(x,y) + \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} tied(x,y) \\
&= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} 1 * |\{r \in R : y \prec_r x\}| + p * |\{r \in R : x \equiv_r y\}| \\
&+ \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} p * (|\{r \in R : x \prec_r y\}| + |\{r \in R : y \prec_r x\}|) \\
&= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \sum_{r \in R} \mathbf{1}_{y \prec_r x} + p * \mathbf{1}_{x \equiv_r y} \\
&+ \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} \sum_{r \in R} p * (\mathbf{1}_{x \prec_r y} + \mathbf{1}_{y \prec_r x}) \\
&= \sum_{r \in R} \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \mathbf{1}_{y \prec_r x} + p * \mathbf{1}_{x \equiv_r y} \\
&+ \frac{1}{2} * \sum_{r \in R} \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} p * (\mathbf{1}_{x \prec_r y} + \mathbf{1}_{y \prec_r x}) \\
&= \sum_{r \in R} |\{(x,y) \in U^2 : x \prec_{r_1} y \wedge y \prec_{r_2} x\}| + p * |\{(x,y) \in U^2 : x \equiv_{r_1} y \wedge x \prec_{r_2} y \vee x \prec_{r_1} y \wedge x \equiv_{r_2} y\}| \\
&= S^p(c,R)
\end{aligned}
$$

Remarque. *La fonction auxiliaire de la KCF correspond ici à la distance de Kendall-τ (Définition 1.2.1) dans le cas où la KCF est équivalente au score de Kemeny (Définition 1.2.2) et à la distance de Kendall-τ généralisée avec pénalité p (Définition 1.3.1) dans le cas où la KCF est équivalente au score de Kemeny généralisé (Définition 1.3.2).*

Propriété 4.2.3. *Le score de Kemeny induit (Définition 2.3.4) et le score de Kemeny étendu (Définition 2.3.8) sont des KCFs.*

Démonstration : Dans la mesure où le score de Kemeny étendu est une généralisation du score de Kemeny induit (le deuxième gère les égalités ce que ne fait pas le premier), nous montrons simplement que le score de Kemeny étendu dont le domaine de définition est $C(U) \times A(U)^{<\infty}$ est une KCF.

Soient $B = (0, 1, 0, 0, 0, 0)$ et $T = (0, 0, 0, 0, 0, 0)$. Soit $(c, R) \in dom(H) = C(U) \times A(U)^{<\infty}$. Nous savons

que :

$$
\begin{aligned}
S^{(B,T)}(c,R) &= \sum_{\substack{x\neq y\in U\\ x\prec_c y}} before(x,y) + \frac{1}{2} * \sum_{\substack{x\neq y\in U\\ x\equiv_c y}} tied(x,y) \\
&= \sum_{\substack{x\neq y\in U\\ x\prec_c y}} 1 * |\{r\in R : y\prec_r x\}| \\
&= \sum_{\substack{x\neq y\in U\\ x\prec_c y}} \sum_{r\in R} \mathbf{1}_{y\prec_r x} \\
&= \sum_{r\in R} \sum_{x\neq y\in U} \mathbf{1}_{x\prec_r y \wedge y\prec_c x} \\
&= \sum_{r\in R} |(x,y)\in (dom(r)\cap dom(c))^2 : x\prec_r y \wedge y\prec_c x| \\
&= H(c,R)
\end{aligned}
$$

Nous observons que les coefficients de T sont tous nuls. En effet, le score de Kemeny étendu ne compte que les désaccords stricts. Cette mesure n'est donc pas pertinente si l'on autorise les égalités dans le classement consensuel : le classement mettant tous les éléments à égalité est nécessairement optimal avec un score de 0 (et le seul cas où il y a un autre classement consensuel optimal est celui où les classements d'entrée sont tous identiques). Par contre, cette mesure peut très bien être utilisée dans le cas où l'on veut forcer que le classement consensuel soit sans égalités (une permutation), ce qui est bien le cas dans [4].

Ainsi, à la place du score de Kemeny étendu, nous considérerons dorénavant le score de Kemeny généralisé induit défini dans les travaux de thèse de Bryan Brancotte [23] qui rassemble le score de Kemeny induit de Dwork *et al.* [44] défini sur les classements incomplets sans égalités et le score de Kemeny généralisé avec paramètre p de Fagin *et al.* [47] défini sur les classements complets avec égalités. La définition de ce score est donnée ci-dessous.

Définition 4.2.5 (Score de Kemeny généralisé induit)**.** *Soit $p\in]0,1]$. Le score de Kemeny généralisé induit entre un classement complet c et une liste de classements R, noté $I^p(c,R)$, est défini de la manière suivante.*

$$
I^p(c,R) = S^{(B,T)}(c,R) \;\; \textit{avec } B=(0,1,p,0,0,0) \textit{ et } T=(p,p,0,0,0,0) \tag{4.3}
$$

Discussion : coefficients et données. En regardant de plus près les coefficients du score de Kemeny généralisé induit, on se rend compte que les pénalités associées aux éléments manquants ($B[4]$ à $B[6]$ et $T[4]$ à $T[6]$) sont toutes nulles. En conséquence, si on a x avant y dans le classement consensuel, il n'y aura aucune pénalité si dans un classement d'entrée on a y présent et x absent. L'absence de x, ici, n'est pas pénalisée vis-à-vis de y. En terme de données, cela veut dire que le score de Kemeny généralisé induit semble à première vue inapproprié s'il

est pertinent de considérer qu'un élément absent d'un classement est moins important ("virtuellement" moins bien classé) qu'un élément présent dans ce même classement.

Reprenons le contexte du *Dataset 1* (reformulations de maladies). Les classements sont ceux renvoyés par la base de données *Gene* du NCBI qui, étant donnée un nom de maladie, renvoie une liste de gènes associés à cette maladie. Dans ce contexte, si un gène est absent d'un classement, il est raisonnable de considérer que la base de donnée a considéré que ce gène est moins pertinent vis-à-vis de la maladie que les gènes retournés. Utiliser le score de Kemeny généralisé induit paraît ici non pertinent.

Reprenons maintenant le deuxième contexte biologique discuté en introduction de ce mémoire de thèse (deuxième cas d'utilisation, page 14). Dans certaines expériences, les conditions expérimentales font que les protéines hydrophobes ne peuvent pas être détectées. Pour autant, leur impact peut être réel. Dans un tel cas, on ne veut pas pénaliser l'absence d'un élément puisque son absence peut être due aux simples conditions expérimentales, sans lien avec la pertinence de l'élément en question vis-à-vis du classement. Dans un tel contexte, l'utilisation du score de Kemeny généralisé induit paraît beaucoup plus pertinent.

Propriété 4.2.4. *Le score de Kemeny généralisé à la pseudo-distance (Définition 2.3.6) est une KCF.*

Démonstration : On montre en utilisant le même principe que pour la preuve de la proposition 4.2.3 et que pour la preuve de la proposition 4.2.2 que le score de Kemeny généralisé à la pseudo-distance est égal à $S^{(B,T)}(c, R)$ avec $B = (0, 1, p, 0, 1, 0)$ et $T = (p, p, 0, p, p, 0)$ pour tout $(c, R) \in dom(H) = C(U) \times A(U)^{<\infty}$. Nous ne donnons donc pas les détails. Le lien entre le score de Kemeny généralisé à la pseudo-distance et les valeurs de B et T est d'ailleurs explicite au regard des définitions 3.3.1, 3.3.2 et 3.3.4.

Discussion : coefficients et données. Cette fois, on peut voir qu'une pénalité de 1 est à payer si l'on veut mettre x avant y dans le classement consensuel alors que dans le classement d'entrée r on a seulement y et pas x. Ce coût est le même que celui d'une inversion : la pénalité est identique que l'on ait y avant x ou y et pas x. On considère donc ici que les éléments présents sont plus importants vis-à-vis du classement d'entrée que les éléments absents.

Nous verrons dans la Section 4.6 que cette pseudo-distance est pertinente pour certains cas d'utilisation comme les reformulations synonymes de maladies, mais pas sur d'autres cas d'utilisation pour lesquels les éléments manquants ne doivent pas être pénalisés vis-à-vis des éléments présents.

Remarque sur la technique d'unification. Pour rappel, la technique d'unification consiste à rendre artificiellement complets des classements qui ne le sont pas en ajoutant à la fin de chaque classement incomplet r_i un *bucket d'unification* correspondant à l'ensemble $U \backslash dom(r_i)$. Cette technique permet alors d'utiliser la distance de Kendall-τ généralisée avec pénalité p définie sur les classements complets.

En fait, utiliser la technique d'unification revient à choisir les vecteurs de pénalités suivants sur les classements d'entrée incomplets :

- $B = (0, 1, p, 0, 1, p)$
- $T = (p, p, 0, p, p, 0)$

En effet, supposons que x soit avant y dans le classement consensuel c. Prenons un classement d'entrée que l'on appelle r. En regardant le vecteur B, on peut faire les observations suivantes :

- La pénalité est identique (et nulle) si x est avant y dans r ou si x est présent et y est absent dans r. C'est en adéquation avec l'unification puisque les éléments absents sont rajoutés à la fin du classement (donc après les éléments présents).
- La pénalité est identique (et de valeur 1) si y est avant x dans r ou si y est présent et x est absent dans r. C'est encore une fois en adéquation avec l'unification pour la même raison qu'au point précédent.
- La pénalité est identique (et de valeur p) si x et y sont à égalité dans r ou si x et y sont tous deux absents de r. C'est en adéquation avec l'unification puisque deux éléments absents se retrouvent à égalité dans le même *bucket d'unification*.

  Un raisonnement analogue avec les valeurs du vecteur T permet de s'assurer que la technique d'unification peut être remplacée par l'utilisation des vecteurs B et T ci-dessus. L'intérêt est de conserver les classements sous leur forme originelle sans avoir à retenir pour chaque classement si son dernier *bucket* est un *bucket d'unification* ou non.

Suite à cette remarque, nous définissons ci-dessous le *score de Kemeny unifiant*.

Définition 4.2.6. *Soit $p \in [0, 1]$. Le score de Kemeny unifiant entre un classement consensuel c et une liste de classements R, noté $F^p(c, R)$, est défini de la manière suivante :*

$$F^p(c, R) = S^{(B,T)} \text{ avec } B = (0, 1, p, 0, 1, p) \text{ et } T = (p, p, 0, p, p, 0). \tag{4.4}$$

Les vecteurs de pénalité des différentes KCFs de la littérature sont rappelés en Table 4.2.

Nom	domaine de définition	vecteur B	vecteur T
Score de Kemeny	$P(U) \times P(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, 0, 0, 0, 0)$	$(0, 0, 0, 0, 0, 0)$
Score de Kemeny généralisé avec pénalité p	$C(U) \times C(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, \mathbf{p}, 0, 0, 0)$	$(\mathbf{p}, \mathbf{p}, \mathbf{0}, 0, 0, 0)$
Score de Kemeny unifiant	$C(U) \times A(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, \mathbf{p}, \mathbf{0}, \mathbf{1}, \mathbf{1})$	$(\mathbf{p}, \mathbf{p}, \mathbf{0}, \mathbf{p}, \mathbf{p}, \mathbf{0})$
Score de Kemeny induit	$P(U) \times Z(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, 0, \mathbf{0}, \mathbf{0}, 0)$	$(0, 0, 0, 0, 0, 0)$
Score de Kemeny généralisé à la pseudo-distance	$C(U) \times A(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, \mathbf{p}, \mathbf{0}, \mathbf{1}, \mathbf{0})$	$(\mathbf{p}, \mathbf{p}, \mathbf{0}, \mathbf{p}, \mathbf{p}, \mathbf{0})$
Score de Kemeny étendu	$P(U) \times A(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$	$(0, 0, 0, 0, 0, 0)$
Score de Kemeny généralisé induit	$C(U) \times A(U)^{<\infty}$	$(\mathbf{0}, \mathbf{1}, \mathbf{p}, \mathbf{0}, \mathbf{0}, \mathbf{0})$	$(\mathbf{p}, \mathbf{p}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$

TABLE 4.2 – Vecteurs de pénalité pour les KCFs de la littérature. Les coefficients en gras sont ceux qui ne sont pas arbitraires compte tenu du domaine de définition de la KCF considérée.

Nous avons montré dans cette sous-section que la classe de KCF englobe les différentes méthodes de l'état de l'art permettant de gérer les classements incomplets avec égalités. Nous avons également donné une première intuition que les coefficients doivent être choisis en fonction du contexte d'application dans la mesure où la présence ou l'absence de pénalité peut s'interpréter en terme de pertinence des éléments absents des classements vis-à-vis des éléments présents.

4.2.3 Classes d'équivalence et complexité

Nous avons vu dans la section précédente que toutes les KCFs n'ont pas le même domaine de définition. Par exemple, le score de Kemeny induit n'est pas défini s'il y a des éléments à égalité dans les classements d'entrée. Le score de Kemeny généralisé avec pénalité p, lui, n'est pas défini si les classements d'entrée sont incomplets. D'autres KCFs sont définies quelques soient les caractéristiques des classements d'entrée. De plus, deux KCFs peuvent être associées à des vecteurs de pénalités très proches. Par exemple, le score de Kemeny adapté à la pseudo-distance (Définition 2.3.6) et le score de Kemeny unifiant (Définition 4.2.6) ne diffèrent que d'un coefficient sur les douze.

L'objectif de cette sous-section est de mettre en évidence des classes d'équivalences liées aux vecteurs de pénalité. Nous allons ainsi établir des conditions suffisantes sur les paramètres du modèle permettant de savoir que deux KCFs sont égales entre elles à une constante multiplicative près sur un sous-ensemble de leur domaine de définition. Nous nous intéresserons plus précisément aux conditions pour que deux KCFs soient équivalentes lorsqu'on ne considère que des permutations et lorsque l'on ne considère que des classements complets.

Equivalence des KCFs

Nous donnons pour commencer une définition d'équivalence pour deux KCFs.

Définition 4.2.7. *Deux KCFs f et g sont équivalentes sur $\mathscr{X} \times \mathscr{Y}^{<\infty}$ avec $\mathscr{X}$ pouvant être $P(U)$ ou $C(U)$ et $\mathscr{Y}$ pouvant être $P(U)$, $C(U)$ ou $A(U)$ si et seulement s'il existe $k \in \mathbb{R}$ tel que $f(c, R) = k * g(c, R)$, pour tout $(c, R) \in \mathscr{X} \times \mathscr{Y}^{<\infty}$.*

Question : A quelles conditions sur les vecteurs de pénalités deux KCFs sont équivalentes lorsque R est une liste de permutations de U et que le classement consensuel doit être une permutation de U ?

Dans le cas des permutations, il est intéressant de voir que toutes les KCFs sont identiques à une constante multiplicative près et sont donc équivalentes. Elles sont donc en particulier toutes équivalentes au score de Kemeny.

Propriété 4.2.5. *Toutes les KCFs sont équivalentes sur $P(U) \times P(U)^{<\infty}$.*

Démonstration : Soient $f = S^{(B,T)}$ et $g = S^{(B',T')}$ deux KCFs. Soit $k = \frac{B'[2]}{B[2]}$. Soit $R \in P(U)^{<\infty}$, autrement dit les classements de R sont des permutations de U. On a alors $\Omega^R_{x,y}[i] = 0$ pour tout $i \geq 3$. Par ailleurs, en utilisant la Définition 4.2.2, nous obtenons $B[1] = 0$. Par ailleurs, en utilisant l'équation (4.1), on obtient que pour tout $c \in P(U)$:

$$f(c,R) = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} B[2] * \Omega^R_{x,y}[2] + \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} T[1] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]).$$

Puisque $c \in P(U)$, il n'y a pas de $x \neq y \in U$ tels que $x \equiv_c y$. En conséquence,

$$f(c,R) = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} B[2] * \Omega^R_{x,y}[2].$$

De la même manière, pour g, on obtient

$$g(c,R) = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} B'[2] * \Omega^R_{x,y}[2] = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} k * B[2] * \Omega^R_{x,y}[2] = k * f(c,R),$$

Intuitivement, si c est une permutation de U, alors il n'y a aucune paire d'éléments à égalité. Ainsi, le vecteur T ainsi que les coefficients $B[3]$ et $B[6]$ deviennent inutiles pour le calcul de $S^{(B,T)}(c,R)$. De plus, les classements étant complets, les coefficients $B[4]$, $B[5]$ et $B[6]$ n'interviennent pas non plus dans le calcul de $S^{(B,T)}(c,R)$. Il ne reste donc plus que les coefficients $B[1]$ et $B[2]$. Le premier étant nécessairement nul selon la définition d'une KCF, la relation d'équivalence devient naturelle.

Une conséquence directe de la propriété ci-dessous est que si R est une liste de permutations, alors deux KCFs distinctes auront le même ensemble de ℓ-consensus optimaux.

Corollaire 4.2.1. *Soient f et g deux KCFs. Si $R \in P(U)^{<\infty}$, alors $\Lambda_f = \Lambda_g$.*

Si les classements d'entrée sont des permutations, alors deux KCFs auront nécessairement le même ensemble de ℓ-consensus optimaux. Par contre, l'ensemble des consensus optimaux peuvent être différents. Prenons un cas simple où $R = [r_1, r_2, r_3]$ avec $r_1 = r_2 = [\{A\}, \{B\}]$ et $r_3 = [\{B\}, \{A\}]$. Il y a un unique ℓ-consensus optimal pour R à savoir $[\{A\}, \{B\}]$. Par contre, si la première KCF a valeur de $T[1] = 1$ et une deuxième KCF a une valeur de $T[1] = 0$, la première a $[\{A\}, \{B\}]$ comme unique classement consensuel optimal alors que la deuxième a $[\{A, B\}]$ comme unique classement consensuel optimal.

Question : A quelles conditions sur leurs vecteurs de pénalités deux KCFs sont équivalentes lorsque R est une liste de classements complets de U ?

Il est maintenant naturel de se demander à quelles conditions deux KCFs sont équivalentes sur $C(U) \times C(U)^{<\infty}$.

Comme pour le cas des permutations, intuitivement, il faut une relation de proportionnalité entre les vecteurs de pénalité des deux KCFs sur les coefficients qui jouent un rôle dans le calcul du score $S^{(B,T)}(c, R)$.

Propriété 4.2.6. *Soient $f = S^{(B,T)}$ et $g = S^{(B',T')}$ deux KCFs. f et g sont équivalentes sur $C(U) \times C(U)^{<\infty}$ si et seulement si il existe $k \in \mathbb{R}$ tel que $B[2] = k * B'[2]$, $B[3] = k * B'[3]$ et $T[1] = k * T'[1]$.*

Démonstration : Soit $k \in \mathbb{R}$ tel que $B[2] = k * B'[2]$, $B[3] = k * B'[3]$ et $T[1] = k * T'[1]$. Soit $R \in C(U)^{<\infty}$. Les classements de R sont donc complets. Autrement dit, $\Omega^R_{x,y}[i] = 0$ pour tout $i \geq 4$.

Grâce à la Définition 4.2.2, nous obtenons $B[1] = T[3] = 0 = B'[1] = T'[3]$, $T[1] = T[2]$ et $T'[1] = T'[2]$. De plus, avec l'équation (4.1), on sait que pour tout $c \in C(U)$, on a :

$$
\begin{aligned}
f(c,R) &= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} B[2] * \Omega^R_{x,y}[2] + B[3] * \Omega^R_{x,y}[3] \\
&\quad + \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} T[1] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]) \\
&= \sum_{\substack{x \neq y \in U \\ x \prec_c y}} k * B'[2] * \Omega^R_{x,y}[2] + k * B'[3] * \Omega^R_{x,y}[3] \\
&\quad + \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} k * T'[1] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]) \\
&= k * \Big(\sum_{\substack{x \neq y \in U \\ x \prec_c y}} B'[2] * \Omega^R_{x,y}[2] + B'[3] * \Omega^R_{x,y}[3] \\
&\quad + \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} T'[1] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]) \Big) \\
&= k * g(c,R).
\end{aligned}
$$

On obtient que f et g sont équivalentes sur $C(U) \times C(U)^{<\infty}$.

Supposons maintenant que f et g sont équivalentes sur $C(U) \times C(U)^{<\infty}$. Alors, il existe $k \in \mathbb{R}$ tel que $f(c, R) = k * g(c, R)$ pour tout $(c, R) \in C(U) \times C(U)^{<\infty}$.

Soit $c_1 = [\{x\}, \{y\}]$, $c_2 = [\{x, y\}]$, $R_1 = [c_1]$, $R_2 = [[\{y\}, \{x\}]]$ et $R_3 = [c_2]$ (R_2 contient un seul classement qui est $[\{y\}, \{x\}]$). On a :

- $f(c_1, R_2) = B[2]$ et $g(c_1, R_2) = B'[2]$
- $f(c_1, R_3) = B[3]$ et $g(c_1, R_3) = B'[3]$
- $f(c_2, R_1) = T[1]$ et $g(c_2, R_1) = T'[1]$

Alors, $B[2] = k * B'[2]$, $B[3] = k * B'[3]$ et $T[1] = k * T'[1]$.

Question : A quelles conditions sur les vecteurs de pénalités d'une KCF avons-nous la garantie que les classements consensuels optimaux d'une liste de permutations de U sont des permutations de U ?

Nous présentons maintenant une condition suffisante sur les vecteurs de pénalités de la KCF permettant d'assurer que si les classements d'entrée sont des permutations de U, alors les classements consensuels optimaux de R sont également des permutations de U.

Propriété 4.2.7. *Soit $f = S^{(B,T)}$ une KCF telle que*

$$T[2] > \frac{B[2]}{2}.$$

Si $R \in P(U)^{<\infty}$, alors $M_f \subseteq P(U)$.

Démonstration : Nous allons montrer que pour tout $c \in C(U) \backslash P(U)$ (c est donc un classement avec au moins une paire d'éléments à égalité), il est impossible que c soit un classement consensuel optimal. Soient $l_1, l_2 \in P(U)$ deux permutations de U telles que, pour tout $x \neq y \in U$, les conditions suivantes soient respectées :

- $x \prec_c y \Rightarrow x \prec_{l_1} y$ et $x \prec_{l_2} y$.
- $x \equiv_c y \Rightarrow (l_1(x) - l_1(y)) * (l_2(x) - l_2(y)) < 0$

(l'ordre relatif de x et y est inversé entre l_1 et l_2 si x et y sont à égalité dans c). Il est facile d'obtenir l_1 et l_2 : pour chaque *bucket* de c contenant au moins deux éléments, on choisit un ordre linéaire arbitraire pour l_1 et son miroir pour l_2.

Nous allons montrer que soit l_1 soit l_2 représente mieux R que c vis-à-vis de f.

Soit $\Delta = 2 * f(c, R) - f(l_1, R) - f(l_2, R)$. Alors, par construction de l_1 et l_2,

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} 2 * tied(x, y) - before(x, y) - before(y, x)$$

Notons que $before(y, x) = \langle (B[2], B[1], B[3], B[5], B[4], B[6]), \Omega^R_{x,y} \rangle$. De plus, comme $R \in P(U)^{<\infty}$, on a $\Omega^R_{x,y}[i] = 0$ pour tout $i \geq 3$, nous obtenons donc :

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} \Omega^R_{x,y}[1] * (2T[1] - B[1] - B[2]) + \Omega^R_{x,y}[2] * (2T[2] - B[1] - B[2])$$

Puisque f est une KCF, nous savons que $B[1] = 0$ et $T[1] = T[2]$.

En définissant $N = |\{x \neq y \in U : x \equiv_c y \text{ et } x \prec_{l_1} y\}|$ nous obtenons :

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} 2\,T[2] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]) - B[2](\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2])$$

$$= N * (2\,T[2] * |R| - B[2] * |R|) = N * |R| * (2T[2] - B[2]) > 0.$$

En conclusion, $f(c, R) > f(l_1, R)$ ou $f(c, R) > f(l_2, R)$, ce qui montre bien que $c \notin M_f$.

Il est également possible de montrer que dans le cas particulier (considéré par Fagin dans [47] pour les classements complets avec égalités) où $T[2] = \frac{B[2]}{2}$, alors il existe au moins un classement consensuel optimal qui soit une permutation de U.

Propriété 4.2.8. *Soit $f = S^{(B,T)}$ une KCF telle que*

$$T[2] = \frac{B[2]}{2}.$$

Si $R \in P(U)^{<\infty}$, alors il existe $\mu \in M_f$ tel que $\mu \in P(U)$.

Démonstration : Nous allons montrer que pour tout $c \in C(U) \backslash P(U)$ (c est donc un classement avec au moins une paire d'éléments à égalité) tel que c est un classement consensuel optimal, il est possible de construire un classement consensuel optimal qui soit une permutation de U. Soient $l_1, l_2 \in P(U)$ deux permutations de U telles que, pour tout $x \neq y \in U$, les conditions suivantes soient respectées :

- $x \prec_c y \Rightarrow x \prec_{l_1} y$ et $x \prec_{l_2} y$.
- $x \equiv_c y \Rightarrow (l_1(x) - l_1(y)) * (l_2(x) - l_2(y)) < 0$

(l'ordre relatif de x et y est inversé entre l_1 et l_2 si x et y sont à égalité dans c). Il est facile d'obtenir l_1 et l_2 : pour chaque *bucket* de c contenant au moins deux éléments, on choisit un ordre linéaire arbitraire pour l_1 et son miroir pour l_2.

Nous allons montrer que soit l_1 soit l_2 représente R aussi bien que c vis-à-vis de f.

Soit $\Delta = 2 * f(c, R) - f(l_1, R) - f(l_2, R)$. Alors, par construction de l_1 et l_2,

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} 2 * tied(x, y) - before(x, y) - before(y, x)$$

Notons que $before(y, x) = \langle (B[2], B[1], B[3], B[5], B[4], B[6]), \Omega^R_{x,y} \rangle$. De plus, comme $R \in P(U)^{<\infty}$, on a $\Omega^R_{x,y}[i] = 0$ pour tout $i \geq 3$, nous obtenons donc :

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} \Omega^R_{x,y}[1] * (2T[1] - B[1] - B[2]) + \Omega^R_{x,y}[2] * (2T[2] - B[1] - B[2])$$

Puisque f est une KCF, nous savons que $B[1] = 0$ et $T[1] = T[2]$.

En définissant $N = |\{x \neq y \in U : x \equiv_c y \text{ et } x \prec_{l_1} y\}|$ nous obtenons :

$$\Delta = \sum_{\substack{x \neq y \in U \\ x \equiv_c y \\ x \prec_{l_1} y}} 2\,T[2] * (\Omega^R_{x,y}[1] + \Omega^R_{x,y}[2]) - B[2](\Omega^R_{x,y}[2] + \Omega^R_{y,x}[2])$$

$$= N * (2\,T[2] * |R| - B[2] * |R|) = N * |R| * (2T[2] - B[2]) = 0.$$

Or, comme c est un classement consensuel optimal par hypothèse, on a $f(c, R) \leq f(l_1, R)$ et $f(c, R) \leq f(l_2, R)$. En conclusion, $f(c, R) = f(l_1, R)$ et $f(c, R) = f(l_2, R)$. Ceci montre bien que l_1 et l_2, permutations de U, sont des classements consensuels optimaux.

Conséquences des propriétés 4.2.7 et 4.2.8 sur la complexité du problème.

Une conséquence de la Propriété 4.2.7 est que le problème d'agrégation de classements incomplets est NP-difficile dès lors que la KCF considérée respecte la condition $T[2] > \frac{B[2]}{2}$. Cependant, certaines KCF ne sont pas définies sur les classements avec égalités, et donc le coefficient $T[2]$ n'est pas défini de manière unique (il ne sera jamais utilisé). Nous obtenons le corollaire ci-dessous.

Corollaire 4.2.2. *Soit f une KCF de domaine de définition $dom(f)$. S'il existe $g = S^{(B,T)}$ une KCF telle que les conditions suivantes sont réunies :*

- *g équivalente à f sur $dom(f)$*
- *$T[2] > \frac{B[2]}{2}$*

alors calculer un classement consensuel optimal qui minimise f est un problème NP-difficile.

Démonstration : Supposons que $T[2] > \frac{B[2]}{2}$. Si $R \in P(U)^{<\infty}$, alors nous savons d'après la Propriété 4.2.7 que tout classement consensuel optimal est une permutation de U. Ainsi, en utilisant la proposition 4.2.3, le problème est identique à celui de minimiser le score de Kemeny qui est connu pour être NP-difficile [12, 20, 44].

Illustration 4.2.1. *Soit $f = S^{(B,T)}$ une KCF définie sur $P(U) \times A(U)^{<\infty}$ avec $B = (0, 1, 0, 0, 0, 0)$ et $T = (0, 0, 0, 0, 0, 0)$. Rien n'indique que le problème d'agrégation de classements soit NP-difficile puisque nous n'avons pas $T[2] > \frac{B[2]}{2}$. Pour autant, le domaine de définition de f indique que le classement consensuel renvoyé est nécessairement*

une permutation de U. f est donc équivalente à $g = S^{(B',T')}$ sur le même ensemble de définition avec $B' = (0,1,0,0,0,0)$ et $T' = (100,100,0,0,0,0)$ (les coefficients $T'[1]$ et $T'[2]$ ne seront jamais utilisés en pratique puisque le classement consensuel n'a pas d'égalités). Ainsi, trouver des classements consensuels optimaux minimisant f est bien NP-difficile.

Corollaire 4.2.3. *Soient $f = S^{(B,T)}$ et $g = S^{(B',T')}$ deux KCFs telles que $T[2] > \frac{B[2]}{2}$ et $T'[2] > \frac{B'[2]}{2}$ Si $R \in P(U)^{<\infty}$, alors $M_f = M_g$.*

Démonstration : On sait par la proposition 4.2.7 que les classements consensuels optimaux de f et g sont des permutations, autrement dit $M_f = \Lambda_f$ et $M_g = \Lambda_g$. Par le Corollaire 4.2.1, on obtient que $\Lambda_f = \Lambda_g$. En définitive, $M_f = M_g$.

Nous avons mis en évidence dans cette sous-section des classes d'équivalence permettant de savoir si deux KCF différentes peuvent retourner la même liste de classements consensuels optimaux selon les caractéristiques des classements d'entrée. Ces classes d'équivalence ont également permis d'établir une condition suffisante pour que le problème d'agrégation de classements soit NP-difficile. Ce résultat montre l'intérêt de développer des algorithmes (i) capables de gérer n'importe quelle KCF et (ii) efficaces pour obtenir rapidement un classement consensuel de bonne qualité.

4.3 Algorithmes pour l'agrégation de classements incomplets

Le but de cette section est de présenter des algorithmes utilisables dans le modèle présenté dans la Section 4.2. Nous commençons par présenter un algorithme exact de type Programmation Linéaire en Nombre Entiers (PLNE) compatible avec n'importe quelle KCF. Nous déclarons ensuite que les méthodes de partitionnement capables de diviser le problème initial en sous-problèmes plus petits présentés dans [7, 19] peuvent être généralisées pour n'importe quelle KCF. Enfin, nous présentons des généralisations de certaines heuristiques classiquement utilisées dans le contexte des classements complets au cadre des KCFs.

4.3.1 Un algorithme exact en PLNE

Nous avons conçu un algorithme exact en PLNE compatible avec n'importe quelle KCF $f = S^{(B,T)}$. L'algorithme prend en entrée une liste de classements ainsi que les vecteurs de pénalité (B,T) et renvoie un classement consensuel optimal vis-à-vis de f.

Dans un premier temps, nous calculons pour toutes les paires (x,y) d'éléments les valeurs de $before(x,y)$, $before(y,x)$ et $tied(x,y)$. La partie PLNE de l'algorithme est présentée ci-dessous.

$$
\begin{array}{rl}
\text{minimiser} & \sum_{x<y} before(x,y) * b_{x,y} + before(y,x) * b_{y,x} + tied(x,y) * t_{x,y} \\
\text{sous les contraintes} & b_{x,y} \in \{0,1\},\, b_{y,x} \in \{0,1\},\, t_{x,y} \in \{0,1\},\, \forall x < y \\
& b_{x,y} + b_{y,x} + t_{x,y} = 1,\, \forall x < y \\
& b_{x,y} + b_{y,z} + t_{min(y,z),max(y,z)} - b_{x,z} \leq 1,\, \forall x \neq y \neq z \neq x \\
& b_{x,y} + t_{min(x,y),max(x,y)} + b_{y,z} - b_{x,z} \leq 1,\, \forall x \neq y \neq z \neq x \\
& t_{x,y} + t_{y,z} - t_{x,z} \leq 1,\, \forall x < y < z
\end{array}
$$

La première contrainte force les variables $b_{x,y}$, $b_{y,x}$ et $t_{x,y}$ à être pseudo-booléennes. Leur valeur est de 1 si et seulement si, respectivement, x est avant y , y est avant x et x est à égalité avec y dans le classement consensuel.

La deuxième contrainte stipule que dans le classement consensuel c, pour une paire donnée (x, y), soit x est avant y, soit y est avant x, soit x et y sont à égalité.

Enfin, les trois dernières contraintes correspondent à des contraintes de transitivité vis-à-vis du classement consensuel qu'on cherche à calculer. Ces contraintes explicitées ci-dessous sont valables pour tout x, y, $z \in U$:

- La troisième contrainte indique que si x est avant y et que y est avant z ou à égalité avec z, alors x est avant z.
- La quatrième contrainte indique que si x est avant y ou à égalité avec y et que y est avant z alors x est avant z.
- La cinquième contrainte indique que si x est à égalité avec y et que y est à égalité avec z, alors x est à égalité avec z [1].

4.3.2 Méthodes de partitionnement

Étant donné la complexité du problème, en pratique, les algorithmes exacts ne peuvent pas être utilisés s'il y a plus de quelques dizaines d'éléments dans U. Comme nous l'avons vu dans le Chapitre 3, il est possible pour la KCF utilisée dans [24] d'utiliser des méthodes basées sur des graphes pour diviser le problème initial en sous-problèmes, calculer un classement consensuel optimal des sous-problèmes puis les concaténer pour obtenir un classement consensuel optimal pour le problème initial. En fait, le *graphe des éléments* défini dans le chapitre précédent (Définition 3.4.1) peut être généralisé pour n'importe quelle KCF.

Définition 4.3.1 (G_e, graphe des éléments dans le contexte d'une KCF)**.** *Soit $G_e = (V_e, A_e)$ le graphe orienté tel que :*

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

La différence entre cette définition de G_e et celle présentée dans le chapitre précédent est que la définition de $before(x, y)$ et $tied(x, y)$ a changé. En effet, dans le chapitre précédent, $before(x, y)$ et $tied(x, y)$ ne dépendaient que du paramètre réel p du modèle de Fagin *et al.* [47] tandis qu'ici $before(x, y)$ et $tied(x, y)$ dépendent des vecteurs

1. $t_{x,y}$ n'est défini que si $x < y$, d'où le fait que l'on impose $x < y < z$.

de pénalité B et T de la KCF.

De la même manière, on peut généraliser la définition d'arc robuste (voir Définition 3.6.3) pour n'importe quelle KCF.

Définition 4.3.2 (Arc robuste dans le contexte d'une KCF)**.** *Soit $G_e = (V_e, A_e)$ le graphe des éléments. On dira que l'arc $(x, y) \in A_e$ est robuste si et seulement si $tied(x, y) > min(x, y)$* [1]*.*

La proposition ci-dessous est toujours valable dans le contexte des KCFs :

Propriété 4.3.1. *Soit R une liste de classements, G_e le graphe des éléments, G_c le graphe des composantes fortement connexes de G_e (voir Définition 3.4.2) et $T = [T_1, ..., T_k]$ un tri topologique de G_c (voir Sous-section 3.4.2). Soit μ_i un classement consensuel optimal pour $R(T_i)$ (pour rappel, $R(T_i)$ correspond à la projection de R sur T_i). La concaténation $\mu_1.\mu_2.....\mu_k$ est un classement consensuel optimal pour R.*

De la même manière que pour la propriété ci-dessus, le théorème des arcs robustes (Théorème 3.6.1) reste valable dans le cadre des KCFs.

Théorème 4.3.1 (Théorème des arcs robustes)**.** *Soient $G_e = (V_e, A_e)$ le graphe des éléments, $\mathscr{R}_e$ l'ensemble des arcs robustes de G_e et $P = [P_1, P_2, ..., P_k]$ une partition ordonnée de V_e telle que*

- *[1.] $\forall i < j$, $\forall x \in P_i$, $\forall y \in P_j$, $(y, x) \notin A_e$, et*
- *[2.] $\forall i$, $\forall x \in P_i$, $\forall y \in P_{i+1}$, $(x, y) \in \mathscr{R}_e$.*

Alors tous les classements consensuels optimaux respectent P.

Les preuves de la Propriété 4.3.1 et du Théorème 4.3.1 sont exactement les mêmes que celles de la proposition 3.5.1 et du Théorème 3.6.1. A nouveau, seules les définitions de $before(x, y)$ et $tied(x, y)$ ont changé. Intuitivement, si les preuves sont identiques alors que la définition de $before(x, y)$ et $tied(x, y)$ ont changé, c'est parce que les preuves de la proposition 3.5.1 et du Théorème 3.6.1 ont été rédigés avec des métafonctions $before(x, y)$ et $tied(x, y)$ correspondant à la KCF utilisée tout au long du Chapitre 3, mais en n'utilisant que des arguments valables quelque soit la KCF utilisée.

Nous montrerons dans le chapitre suivant que l'utilisation de cette méthode de partitionnement peut réduire drastiquement le temps de calcul d'un classement consensuel optimal. Cependant, si l'une des composantes fortement connexes de G_c contient plus de quelques dizaines d'éléments, il n'est pas possible d'utiliser un algorithme exact pour le sous-problème associé en raison de sa taille. Dans cette situation, l'utilisation d'une heuristique est alors nécessaire. Il est donc important de proposer des heuristiques capables de fonctionner sur les KCFs.

1. Le fait que (x, y) soit un arc de G_e implique déjà que $before(y, x) > min(x, y)$. On a donc que (x, y) est robuste si et seulement si le coût de placer x avant y dans le classement consensuel est strictement inférieur au coût de n'importe quel autre positionnement relatif pour la paire (x, y).

4.3.3 Adaptation des heuristiques de Kemeny pour les KCFs

Plusieurs heuristiques sont classiquement utilisées dans le cadre de l'agrégation de permutations. Plusieurs d'entre elles comme KwikSort [3], CopelandMethod [36], BioConsert [32], Borda [39], MedRank [46], Pick -A-Perm [3] ont été généralisées dans [25] pour gérer les classements avec égalités. Parmi elles, certaines sont facilement adaptables pour gérer n'importe quelle KCF (et donc une véritable diversité de contextes possibles pour les données manquantes) tandis que d'autres ne le sont pas. Fondamentalement, les heuristiques basées sur des comparaisons par paires peuvent facilement être généralisées aux KCFs.

Généralisation des heuristiques basées sur des comparaisons par paires

KwikSort. KwikSort [3] est une $2-$approximation dans le contexte de l'agrégation de permutations. C'est un algorithme de type diviser pour régner inspiré de l'algorithme de tri rapide (d'où son nom). Il est défini dans [3] et adapté par [25] pour gérer les égalités. Dans l'adaptation de [25], le principe est le suivant : on commence par sélectionner un élément pivot, puis chaque élément restant y doit être placé (i) dans le groupe de gauche si y est avant le pivot dans une majorité de classements, (ii) dans le groupe de droite si le pivot est avant y dans une majorité de classements, et (iii) dans le même groupe que le pivot si le pivot et y sont à égalité dans une majorité de classements. Ensuite, on répète cette procédure de manière récursive sur le groupe de gauche et le groupe de droite. Afin d'adapter KwikSort au contexte des KCFs, il suffit que l'élément y soit dans le groupe de gauche si $before(y, pivot) = min(pivot, y)$, dans le groupe de droite si $before(pivot, y) = min(pivot, y)$ et dans le même groupe que le pivot si $tied(pivot, y) = min(pivot, y)$. Ainsi, seront dans le groupe de gauche les éléments dont le coût d'être positionnés avant le pivot dans le classement consensuel est minimal, dans le groupe de droite les éléments dont le coût d'être positionnés après le pivot dans le classement consensuel est minimal, et enfin dans le même groupe que le pivot les éléments dont le coût d'être positionnés à égalité avec le pivot dans le classement consensuel est minimal. Le pseudo-code est donné dans l'Algorithme 3.

Méthode Copeland. De la même manière, la méthode Copeland [36] peut être généralisée pour s'adapter à n'importe quelle KCF. Dans sa version originelle, on associe un score à chaque élément. Le score d'un élément x donné est le nombre d'éléments que x "bat en duel" (dans le contexte d'une élection) moins le nombre d'éléments qui battent x en duel. Pour s'adapter au contexte des KCFs, on considère maintenant que le score associé à un élément x est le nombre d'autres éléments y tels que $before(x, y)$ est strictement inférieur à $before(y, x)$ et à $tied(x, y)$ moins le nombre d'éléments y tels que $before(y, x)$ est strictement inférieur à $before(x, y)$ et à $tied(x, y)$. La traduction de "x bat y en duel" dans notre contexte de KCF se traduit par le fait que le coût de mettre x avant y est strictement inférieur au coût de mettre y avant x et x à égalité avec y. Comme dans [25], deux éléments dont le score final est identique se retrouvent à égalité dans le classement consensuel. Le pseudo-code de l'algorithme CopelandMethod dans sa version KCF est donnée dans l'Algorithme 4.

Algorithme 3 : KwikSort généralisé aux KCFs

Input : R : Liste de classements, B : vecteur réel de taille 6, T : vecteur réel de taille 6
Algorithme `KwikSort(`*R : Liste de classements, B et T : vecteurs réels de taille 6*`)`

```
1   consensus ← EmptyListOfSets()
2   U ← SetOfDistinctElementsInR(R)
3   KwikSortAux(U, consensus)
4   return consensus
```

Result : classement consensuel

Procedure `KwikSortAux(`*R : Liste de classements, U : ensemble, consensus : liste d'ensembles, B et T : vecteurs réels de taille 6*`)`

```
1
2   p ← PivotAleatoire(U)
3   gauche ← EmptySet()
4   centre ← EmptySet()
5   droite ← EmptySet()
6   ajouter p à centre
7
8   for y ≠ p ∈ U do
9       beforePY ← before(p, y, B, T)
10      beforeYP ← before(y, p, B, T)
11      tiedPY ← tied(p, y, B, T)
12      if beforeYP ≤ beforePY and beforeYP ≤ tiedPY then
13          ajouter y à gauche
14      else if beforePY ≤ beforeYP and beforePY ≤ tiedPY then
15          ajouter y à droite
16      else
17          ajouter y à centre
        end
    end
18
19  if length(gauche) > 0 then
20      KwikSortAux(R, gauche, consensus, B, T)

21  ajouter centre à la fin de consensus
22
23  if length(droite) > 0 then
24      KwikSortAux(R, droite, consensus, B, T)
```

Algorithme 4 : CopelandMethod généralisé aux KCFs

Input : R : Liste de classements, B : vecteur réel de taille 6, T : vecteur réel de taille 6

Algorithme `CopelandMethod(`*R : Liste de classements, B et T : vecteurs réels de taille 6*`)`

```
1   consensus ← EmptyListOfSets()
2   U ← ListOfDistinctElementsInR(R)
3   S ← Array(size=size(U), values=0)
4   for i ← 1 to size(U) - 1 do
5       for j ← i + 1 to size(U) do
6           (beforeXY, beforeYX, tiedXY) ← computeCostRelativePositions(U[ i ], U[ j ], R, B, T)
8           if beforeXY < beforeYX and beforeXY < tiedXY then
9               S[ i ] = S[ i ] + 1
10              S[ j ] = S[ j ] - 1
11          else if beforeYX < beforeXY and beforeYX < tiedXY then
12              S[ i ] = S[ i ] - 1
13              S[ j ] = S[ j ] + 1
        end
14      trier U selon les valeurs de S
15      bucket ← EmptySet()
16      k ← 0
17      scoreActuel ← S[0]
18      while k < length(U) do
19          if S[k] ≠ scoreActuel then
20              ajouter bucket à consensus
21              bucket ← EmptySet()
22              scoreActuel ← S[k]
23          ajouter U[k] à bucket
24          k = k + 1
        end
25      ajouter bucket à consensus
26      return consensus
    end
```

Result : classement consensuel

BioConsert. BioConsert est une heuristique de type recherche locale publiée dans [32]. Elle a été initialement conçue pour prendre en compte les classements complets avec égalités (basé sur le score de Kemeny généralisé). L'idée est de partir d'un ou plusieurs classements complets (classements d'entrée possiblement unifiés s'ils sont incomplets ou classements consensuels produits par d'autres algorithmes) et d'effectuer un certain nombre de déplacements d'éléments si ces mouvements permettent de diminuer le score de Kemeny (donc d'améliorer le classement consensuel).

Pour un élément $x \in U$, les deux mouvements ci-dessous sont autorisés.

- déplacer x dans un *bucket* déjà existant.
- déplacer x dans un nouveau *bucket*.

A chaque étape, toutes les positions où x peut être déplacé sont testées : s'il y a k *buckets*, il y a $k-1$ possibilités de déplacer x dans un *bucket* existant et $k+1$ positions pour créer un nouveau *bucket*.

L'algorithme s'arrête lorsque plus aucun déplacement autorisé ne permet de diminuer le score de Kemeny.

En pratique, cet algorithme peut être codé selon plusieurs choix d'implémentation selon l'ordre dans lequel les possibilités de déplacement sont explorées.

Le pseudo-code présenté en Algorithme 5 décrit l'implémentation à partir de laquelle les expériences de la Section 4.6 ont été effectuées.

La différence entre la version native de BioConsert, basée sur le score de Kemeny généralisé et la version de BioConsert proposée ici et utilisable sur n'importe quelle KCF est que la recherche de mouvements pertinents dépend des vecteurs de pénalité B et T. Des optimisations ont été proposées dans [23] et s'adaptent parfaitement aux KCFs.

Généralisation d'heuristiques non basées sur des comparaisons par paires

Les méthodes positionnelles comme Borda [39] et MedRank [46], ne peuvent pas être facilement généralisées à n'importe quelle KCF car elles se concentrent sur les positions des éléments à classer dans les classements d'entrée et ne peuvent pas, par essence, gérer l'infinité de valeurs possibles de paramètres pour les KCFs.

Pick-A-Perm Pick-A-Perm [3] consiste à choisir un classement d'entrée et à le renvoyer comme classement consensuel. Selon les variantes, le choix peut être effectué au hasard ou bien on peut choisir le classement d'entrée qui minimise le score de Kemeny avec R.

Cet algorithme ne s'applique pas aux KCFs car les classements d'entrée peuvent être incomplets ce qui n'est pas compatible avec la définition du classement consensuel.

Bien que ces heuristiques ne puissent pas s'adapter à toutes les KCFs, il peut arriver qu'elles puissent s'adapter à certaines KCFs de l'état de l'art. Par exemple, si la KCF est équivalente au processus d'unification (voir Définition

Algorithme 5 : BioConsert généralisé aux KCFs

Input : R : Liste de classements, B : vecteur réel de taille 6, T : vecteur réel de taille 6

Algorithme `BioConsert(`*R : Liste de classements (unifiés si incomplets), B et T : vecteurs réels de taille 6*`)`

```
1   U ← SetOfDistinctElementsInR(R)
2   minScore ← score(R[1], R)
3   for r in R do
4       amelioreClassement (r, U, B, T)
5       scoreR ← score(r, R)
6       if scoreR < minScore then
7           consensus ← r
8           minScore ← scoreR
    end
9   return consensus
```

Procedure `amelioreClassement(`*r : classement, U : ensemble, consensus : liste d'ensembles, B et T : vecteurs réels de taille 6*`)`

```
1
2   deplacementPertinent ← True
3   while deplacementPertinent do
4       deplacementPertinent ← False
5       for x ∈ U do
6           bucketX ← getPositionBucket(x, r)
7           nbBuckets ← getNbBuckets(r)
8           i ← bucketX+1
9           while i <= nbBuckets and not deplacementPertinent do
                if diminueScoreChangementBucket(x, i, B, T) then
10                  changeBucket(x,i)
11                  deplacementPertinent ← True
12              i ← i + 1
            end
13          i ← bucketX-1
14          while i >= 1 and not deplacementPertinent do
                if diminueScoreChangementBucket(x, i, B, T) then
15                  changeBucket(x,i)
16                  deplacementPertinent ← True
17              i ← i - 1
            end
18          i ← bucketX
19          while i <= nbBuckets and not deplacementPertinent do
                if diminueScoreAjoutBucketPourXApres(x, i, B, T) then
20                  ajouteBucketPourXApres(x,i)
21                  deplacementPertinent ← True
22              i ← i + 1
            end
23          i ← bucketX
24          while i >= 1 and not deplacementPertinent do
                if diminueScoreAjoutBucketPourXAvant(x, i, B, T) then
25                  ajouteBucketPourXAvant(x,i)
26                  deplacementPertinent ← True
27              i ← i - 1
            end
        end
    end
```

4.2.6), les classements d'entrée peuvent être unifiés. En conséquence, toutes les heuristiques capables de gérer des classements complets avec égalités sont utilisables.

Borda Nous expliquons maintenant comment Borda [39] peut être généralisé pour la KCF définie par Dwork *et al.* dans [44] c'est-à-dire au score de Kemeny induit (Définition 2.3.4). Cette KCF considère que les éléments manquants ne sont pas comparables avec les éléments présents dans un classement donné. En conséquence, avec cette KCF, un élément n'est pas pénalisé lorsqu'il n'est pas présent dans un classement. Borda consiste à calculer un classement consensuel en fonction de la somme des positions de chaque élément dans les classements d'entrée. Une manière naturelle de généraliser Borda pour cette KCF est de calculer pour chaque élément sa position moyenne en ne considérant que les classements d'entrée dans lesquels l'élément est présent. Ensuite, le classement consensuel est calculé en triant les éléments par ordre croissant du score ainsi obtenu.

MedRank Comme l'heuristique Borda, MedRank [46] peut être utilisée sur des classements incomplets dans certains cas que nous allons expliciter. L'idée de cette heuristique, sur des permutations, est d'associer à chaque élément x un score correspondant au plus petit k tel que x apparaît dans le top-k d'au moins 50% des classements. Dans le classement consensuel final, les éléments sont alors rangés par ordre croissant de score. Cet algorithme a été adapté pour les classements avec égalités dans [25].

Voyons maintenant comment cet algorithme peut être adapté à certaines KCFs. Si la KCF correspond au score de Kemeny unifiant (Définition 4.2.6), il suffit d'appliquer un processus d'unification et d'utiliser l'algorithme décrit dans [25] (généralisation de MedRank aux classements avec égalités). Si la KCF correspond au score de Kemeny induit (Définition 2.3.4), comme pour l'algorithme Borda, il faut adapter la définition du score associé à un élément x. Comme l'absence d'un élément dans un classement ne doit pas porter préjudice à cet élément, nous proposons que le score de x soit le plus petit entier k tel que x est présent dans le top-k d'au moins la moitié du nombre de classements dans lesquels il apparaît.

4.4 Axiomatisation du modèle présenté

Jusqu'ici, le problème d'agrégation de classements a été vu sous le prisme de la méthode Kemeny-Young consistant à chercher le classement consensuel qui minimise le score de Kemeny [59]. Initialement, ce problème a été défini dans un contexte d'élections : l'ensemble U correspond à l'ensemble des candidats à l'élection et les classements d'entrée correspondent aux votes des électeurs tenus de classer l'ensemble des candidats par ordre de préférence. Pour déterminer le résultat de l'élection et en particulier désigner le vainqueur, plusieurs méthodes d'agrégation de classements sont possibles (elles correspondent à des systèmes électoraux). La méthode Borda [39], la méthode Copeland [36], la méthode Schulze [79] sont ainsi des alternatives à la méthode Kemeny-Young

pour calculer un classement consensuel. Ce sont les critères d'équité qui permettent de comparer les méthodes d'agrégation de classements entre-elles. Par exemple, la méthode Borda ne respecte pas le *majority criterion*[1] mais respecte le *consistency criterion*[2] tandis que la méthode Kemeny-Young respecte le *majority criterion* mais ne respecte pas le *consistency criterion*. Dans cette section, nous nous focalisons sur trois critères d'équité particulièrement importants : le *critère de majorité*, le *critère de Condorcet* et l'*indépendance locale des alternatives non pertinentes* (*local independance of irrelevant alternatives*) généralement appelé LIIA. Le critère de majorité et le critère de Condorcet sont intéressants puisqu'ils déterminent chacun une condition suffisante permettant dans le cadre de la méthode Kemeny-Young de partitionner le problème initial en sous-problèmes indépendants. Le critère de Condorcet fait d'ailleurs partie des critères d'équité les plus cités dans la littérature (par exemple, [9, 19, 26, 49]) et est à la base du critère de Condorcet étendu [83] permettant de calculer une partition ordonnée respectée par tous les classements consensuels optimaux. Le LIIA est fondamental puisqu'il assure qu'une condition suffisante permettant de trouver les premiers éléments d'un classement consensuel optimal peut être utilisée récursivement sur les éléments restants. L'objectif de cette section est d'une part d'établir des liens entre le respect de ces critères et les vecteurs de pénalité B et T du modèle présenté au début de ce chapitre (ce problème reste ouvert pour de nombreux autres critères d'équité) et d'autre part de montrer que selon la nature des données, il est possible de vouloir ne pas respecter certains de ces critères.

4.4.1 Lemmes préliminaires

L'objectif de cette sous-section est de présenter et de démontrer un certain nombre de lemmes particulièrement utiles à la section suivante. L'idée générale de ces lemmes est la suivante : étant donnés deux éléments x et y de U et indépendamment des autres éléments de U, est-il plus intéressant vis-à-vis de la KCF à minimiser de positionner x avant y, y avant x ou x à égalité avec y dans le classement consensuel ?

Le premier lemme s'intéresse aux deux questions suivantes :

- quelle est la différence de coût entre le fait de positionner x avant y et le fait de positionner y avant x dans le classement consensuel ?
- quelle est la différence de coût entre le fait de positionner x avant y et le fait de positionner x à égalité avec y dans le classement consensuel ?

Lemme 4.4.1. *Soit $S^{(B,T)}$ une KCF. Alors, pour tout $x, y \in U$ et $R \in A(U)^{<\infty}$, on a :*

$$\begin{aligned} before(x,y) - before(y,x) = B[2] * (\Omega^R_{x,y}[2] - \Omega^R_{x,y}[1]) \\ + (B[5] - B[4]) * (\Omega^R_{x,y}[5] - \Omega^R_{x,y}[4]). \end{aligned} \tag{4.5}$$

1. Un système électoral respecte le *majority criterion* si un candidat qui est premier dans une stricte majorité de classements est nécessairement le vainqueur de l'élection.
2. Un système électoral respecte le *consistency criterion* si pour toute paire (R_1, R_2) de liste de classements de U telle que l'ensemble des vainqueurs pour R_1 est égal à l'ensemble des vainqueurs pour R_2, l'ensemble des vainqueurs de $R_1 \bigcup R_2$ est "conservé".

Par ailleurs, nous avons également :

$$before(x,y) - tied(x,y) = \sum_{1 \leq i \leq 6} \Omega^R_{x,y}[i] * (B[i] - T[i]). \tag{4.6}$$

Démonstration : Par définition de Ω (cf Définition 4.2.1), on a $\Omega^R_{y,x} = (\Omega^R_{x,y}[2], \Omega^R_{x,y}[1], \Omega^R_{x,y}[3], \Omega^R_{x,y}[5], \Omega^R_{x,y}[4], \Omega^R_{x,y}[6])$.

Ainsi, $\Omega^R_{x,y} - \Omega^R_{y,x} = (\Omega^R_{x,y}[1] - \Omega^R_{x,y}[2], \Omega^R_{x,y}[2] - \Omega^R_{x,y}[1], 0, \Omega^R_{x,y}[4] - \Omega^R_{x,y}[5], \Omega^R_{x,y}[5] - \Omega^R_{x,y}[4], 0)$.

De plus, $S^{(B,T)}$ étant une KCF, par définition on a $B[1] = 0$. Nous obtenons donc l'équation 4.5.

L'équation 4.6 n'est qu'une opération algébrique élémentaire.

Il est intéressant de constater que la différence de coût entre le fait de positionner x avant y et le fait de positionner y avant x dans le classement consensuel ne dépend que de trois paramètres réels parmi les douze définissant une KCF : $B[2]$, $B[4]$ et $B[5]$.

Remarque. *On peut remarquer que $tied(x,y) - tied(y,x)$ vaut nécessairement zéro puisque le coût de mettre x à égalité avec y est le même que celui de mettre y à égalité avec x.*

Les deux lemmes suivants présentent des conditions suffisantes sur les vecteurs B et T garantissant que si dans une stricte majorité de classements d'entrée on a $x \prec_r y$ ou $x \diamond_r y$ (x avant y ou x présent et y absent), alors le coût de placer x avant y dans un classement consensuel (qu'il soit optimal ou non) est plus petit (donc plus intéressant) que le coût de placer y avant x ou de mettre x à égalité avec y.

Le premier de ces lemmes donne une condition permettant d'assurer que le coût de positionner un élément x avant un élément y est strictement plus petit que le coût de positionner y avant x.

Lemme 4.4.2. *Soit $f = S^{(B,T)}$ une KCF telle que $B[2] = B[5] - B[4]$ et soit $x \neq y \in U$. Si $\sum_{i \in \{1,4\}} \Omega^R_{x,y}[i] > \sum_{j \in \{2,5\}} \Omega^R_{x,y}[j]$, alors $before(x,y) < before(y,x)$.*

Démonstration : Soit $A = before(x,y) - before(y,x)$. En utilisant (4.5), on obtient :

$$\begin{aligned} A &= B[2] * (\Omega^R_{x,y}[2] - \Omega^R_{x,y}[1]) \\ &\quad + (B[5] - B[4]) * (\Omega^R_{x,y}[5] - \Omega^R_{x,y}[4]) \\ &= B[2] * (\Omega^R_{x,y}[2] - \Omega^R_{x,y}[1] + \Omega^R_{x,y}[5] - \Omega^R_{x,y}[4]) < 0. \end{aligned}$$

Le second de ces lemmes donne une condition permettant d'assurer que le coût de positionner un élément x avant un élément y est strictement plus petit que le coût de positionner x à égalité avec y. L'intuition derrière ce lemme est la suivante : si par exemple les coûts $T[1]$ et $T[4]$ sont suffisamment élevés (on considère ainsi l'ensemble

$P_1 = \{1, 4\}$), le coût de mettre à égalité x et y dans le classement consensuel va augmenter significativement chaque fois que x est avant y ou que x est présent et y absent dans un classement d'entrée. On obtient que si dans une stricte majorité de classements d'entrée on a $x \prec_r y$ ou $x \diamond_r y$, alors, le positionnement relatif le moins onéreux dans le classement consensuel c pour la paire (x, y) est $x \prec_c y$. (Pour autant x ne sera pas forcément devant y dans le classement consensuel optimal, car d'autres éléments situés entre eux peuvent imposer un classement contradictoire). En généralisant à n'importe quel ensemble $P_1 \subseteq \{1, 2, 3, 4, 5, 6\}$, on obtient le lemme suivant :

Lemme 4.4.3. *Soit $f = S^{(B,T)}$ une KCF et soit x et y deux éléments distincts de U. Soit $P = \{1, 2, 3, 4, 5, 6\}$, et considérons P_1 et P_2 deux sous-ensembles disjoints de P.*

Si les conditions suivantes sont vérifiées :

- $\underset{i \in P_1}{min}(T[i] - B[i]) > 0$,
- $\underset{i \in P_1}{min}(T[i] - B[i]) \geq \underset{j \in P_2}{max}(B[j] - T[j])$,
- $\sum_{i \in P_1} \Omega^R_{x,y}[i] > \sum_{j \in P_2} \Omega^R_{x,y}[j]$,
- $\forall k \notin P_1 \cup P_2$, $\Omega^R_{x,y}[k] = 0$,

alors $before(x, y) < tied(x, y)$.

Démonstration : Supposons que $\sum_{i \in P_1} \Omega^R_{x,y}[i] > \sum_{j \in P_2} \Omega^R_{x,y}[j]$. Soit $A = before(x, y) - tied(x, y)$. En utilisant (4.6), on obtient :

$$\begin{aligned} A &= \sum_{i \in P_1} \Omega^R_{x,y}[i] * (B[i] - T[i]) + \sum_{j \in P_2} \Omega^R_{x,y}[j] * (B[j] - T[j]) \\ &= \sum_{j \in P_2} \Omega^R_{x,y}[j] * (B[j] - T[j]) - \sum_{i \in P_1} \Omega^R_{x,y}[i] * (T[i] - B[i]). \end{aligned}$$

Soit $A_1 = \underset{i \in P_1}{min}(T[i] - B[i])$ et $A_2 = \underset{j \in P_2}{max}(B[j] - T[j])$. Alors

$$\sum_{i \in P_1} \Omega^R_{x,y}[i] * (T[i] - B[i]) \geq A_1 * \sum_{i \in P_1} \Omega^R_{x,y}[i]$$

et

$$\sum_{j \in P_2} \Omega^R_{x,y}[j] * (B[j] - T[j]) \leq A_2 * \sum_{j \in P_2} \Omega^R_{x,y}[j].$$

De plus comme $A_1 > 0$, $A_1 \geq A_2$ et $\sum_{i \in P_1} \Omega^R_{x,y}[i] > \sum_{j \in P_2} \Omega^R_{x,y}[j]$, on obtient bien $A < 0$.

Revenons sur l'intuition associée à ce lemme. L'ensemble P comprenant les nombres entiers de 1 à 6 représentant les six possibilités présentées Table 4.1 pour deux éléments x et y dans un classement d'entrée r (x avant y, x après y, ..., x et y absents). Les cas n'apparaissant jamais dans les classements d'entrée ne se retrouvent ni dans P_1, ni dans P_2. A l'inverse, tous les cas représentés au moins une fois dans R doivent être soit dans P_1, soit dans

P_2. Dans P_1, on va avoir les entiers i tels que les $T[i] - B[i]$ sont élevés (les cas défavorables à l'égalité). Si ces cas sont majoritaires dans les classements d'entrée, alors le coût de mettre x à égalité avec y va être supérieur à celui de mettre x avant y.

Les deux lemmes ci-dessous mettent en avant un phénomène intéressant. En effet, ils présentent des conditions suffisantes sur les vecteurs de pénalité qui permettent d'obtenir le comportement suivant : un élément x de U est l'unique premier dans une stricte majorité de classements, mais x n'est unique premier dans aucun des classements consensuels optimaux. En première approche, on pourrait penser que ce comportement n'est pas souhaitable. Cependant dans le cas des données manquantes, on verra par la suite que ce comportement peut être souhaitable ou non, suivant l'interprétation des données manquantes.

Ces deux lemmes sont énoncés dans le cas particulier où l'ensemble U n'a que deux éléments. Ils serviront à construire des contre-exemples pour montrer que certains critères ne sont pas respectés.

Lemme 4.4.4. *Supposons que U contienne uniquement deux éléments. Autrement dit, on considère que $U = \{X, Y\}$. Soit $f = S^{(B,T)}$ une KCF telle que $B[2] \neq B[5] - B[4]$. Alors, il existe $R = [r_1, \ldots, r_m] \in A(U)^{<\infty}$ tel que*

- *$\Omega^R_{X,Y}[3] = \Omega^R_{X,Y}[6] = 0$, en particulier, X et Y ne sont jamais à égalité,*
- *$\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}$,*
- *$[\{X\}, \{Y\}]$ n'est pas un classement consensuel optimal de R vis-à-vis de f.*

Démonstration :

Comme $U = \{X, Y\}$, il y a seulement trois classements consensuels possibles : $c_1 = [\{X\}, \{Y\}]$, $c_2 = [\{Y\}, \{X\}]$ et $c_3 = [\{X, Y\}]$. En utilisant la Définition 4.2.1, $f(c_1, R) = before(x, y)$ et $f(c_2, R) = before(y, x)$.

Supposons que $B[2] \neq B[5] - B[4]$. Isolons les deux cas possibles.

Cas 1 : $B[2] < B[5] - B[4]$. Nous construisons R de sorte que R contienne $[\{X\}, \{Y\}]$ $t + 1$ fois et $[\{Y\}]$ t fois, alors

$$\Omega^R_{X,Y} = (t + 1, 0, 0, 0, t, 0).$$

Par construction, $m = 2t + 1$ et

$$\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}.$$

Soit $A_1 = before(x, y) - before(y, x) = f(c_1, R) - f(c_2, R)$. En utilisant l'équation (4.5), nous obtenons que

$$\begin{aligned} A_1 &= -B[2] * (t + 1) + t(B[5] - B[4]) \\ &= t(B[5] - B[4] - B[2]) - B[2]. \end{aligned}$$

Donc $A_1 > 0$ est équivalent à $t > \frac{B[2]}{B[5] - B[4] - B[2]}$. Ainsi pour t assez grand, c_1 n'est pas optimal pour R.

Cas 2 : $B[2] > B[5] - B[4]$. Nous construisons alors R de sorte que R contienne $[\{X\}]$ $t+1$ fois et $[\{Y\}, \{X\}]$ t fois, alors

$$\Omega^R_{X,Y} = (0, t, 0, t+1, 0, 0).$$

Par construction,

$$\sum_{i \in \{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}.$$

Soit $A_2 = before(x, y) - before(y, x)$. A nouveau, par l'équation (4.5), nous obtenons

$$\begin{aligned} A_2 &= t * B[2] + (B[5] - B[4]) * (-t - 1) \\ &= t(B[2] + B[4] - B[5]) + B[4] - B[5]. \end{aligned}$$

Donc $A_2 > 0$ est équivalent à $t > \frac{B[5] - B[4]}{B[2] + B[4] - B[5]}$. Ainsi pour t assez grand, c_1 n'est pas optimal. (Quand $B[5] < B[4]$, on peut prendre n'importe quelle valeur positive pour t).

Dans les deux cas, on a pu construire une liste de classements R vérifiant les conditions du lemme.

Une adaptation du score de Kemeny couramment utilisée répond aux conditions initiales du lemme : il s'agit du score de Kemeny induit (Définition 2.3.4) associé à la mesure de Kendall-τ induite (Définition 2.3.3) définie par Dwork *et al.* dans [44]. Nous avons vu dans la Sous-section 4.2.2 que cette adaptation du score de Kemeny est pertinente lorsqu'on ne veut pas considérer qu'un élément absent d'un classement donné est moins pertinent vis-à-vis de ce classement que les éléments qui y sont présents. Ainsi, si un élément x est premier dans une stricte majorité de classements mais que y est absent de ces classements (incomparables) et que dans les autres classements c'est y qui est avant x, alors on veut qu'à la fin ce soit y qui se retrouve avant x dans les classements consensuels optimaux.

Le lemme ci-dessous est dans la même idée que le précédent : Si on prend $P_1 = \{1, 4\}$ dans ce lemme, on a un élément x qui est l'unique premier dans une stricte majorité de classements mais qui n'est unique premier dans aucun des classements consensuels optimaux.

Lemme 4.4.5. *Soit $P = \{1, 2, 3, 4, 5, 6\}$, et P_1 et P_2 deux sous-ensembles disjoints de P. Supposons $U = \{X, Y\}$, et soit $f = S^{(B,T)}$ une KCF telle que $\min_{i \in P_1}(T[i] - B[i]) < \max_{j \in P_2}(B[j] - T[j])$. Alors il existe $R \in A(U)^{<\infty}$ tel que*

- $\sum_{i \in P_1} \Omega^R_{X,Y}[i] > \sum_{j \in P_2} \Omega^R_{X,Y}[j]$,
- $\forall k \notin P_1 \cup P_2$, $\Omega^R_{X,Y}[k] = 0$, *et*
- $[\{X\}, \{Y\}]$ *n'est pas un classement consensuel optimal de R vis-à-vis de f.*

Démonstration : A nouveau, les trois seuls classements consensuels possibles sont $c_1 = [\{X\}, \{Y\}]$, $c_2 = [\{Y\}, \{X\}]$ et $c_3 = [\{X, Y\}]$.

En utilisant la Définition 4.2.1, on obtient

$$f(c_1, R) = before(x, y) \quad \text{and} \quad f(c_3, R) = tied(x, y).$$

Les classements possibles sur U sont :

$$\begin{aligned} r_1 &= [\{X\}, \{Y\}], \quad r_2 = [\{Y\}, \{X\}], \quad r_3 = [\{X, Y\}], \\ r_4 &= [\{X\}], \quad r_5 = [\{Y\}] \quad \text{et} \quad r_6 = [\,]. \end{aligned}$$

Donc pour toute liste de classements R, pour tout i, $\Omega^R_{X,Y}[i]$ est le nombre de fois que le classement r_i est dans R.

Prenons $\alpha_1 \in P_1$ tel que $T[\alpha_1] - B[\alpha_1] = \underset{i \in P_1}{min}(T[i] - B[i])$, et prenons $\alpha_2 \in P_2$ tel que $T[\alpha_2] - B[\alpha_2] = \underset{j \in P_2}{max}(B[j] - T[j])$.

Nous construisons alors R afin qu'il contienne t fois r_{α_2} et $t + 1$ fois r_{α_1}. Par construction, on obtient que

$$\sum_{i \in P_1} \Omega^R_{X,Y}[i] = t + 1 \ > \ t = \sum_{j \in P_2} \Omega^R_{X,Y}[j],$$

et que $\Omega^R_{X,Y}[k] = 0$ pour les $k \notin P_1 \cup P_2$.

Posons $A = before(x, y) - tied(x, y) = f(c_1, R) - f(c_3, R)$. Par l'équation (4.6), on obtient que

$$\begin{aligned} A &= \Omega^R_{X,Y}[\alpha_1] * \underset{i \in P_1}{min}(T[i] - B[i]) + \Omega^R_{X,Y}[\alpha_2] * \underset{j \in P_2}{max}(B[j] - T[j]) \\ &= (t + 1) * \underset{i \in P_1}{min}(T[i] - B[i]) + t * \underset{j \in P_2}{max}(B[j] - T[j]). \end{aligned}$$

Donc

$$A > 0 \ \Leftrightarrow \ t > \frac{-\underset{i \in P_1}{min}(T[i] - B[i])}{\underset{i \in P_1}{min}(T[i] - B[i]) + \underset{j \in P_2}{max}(B[j] - T[j])}.$$

Ainsi pour t assez grand, c_1 n'est pas optimal pour R.

4.4.2 Définitions préliminaires

Dans les sections suivantes, nous reprenons plusieurs critères d'équité fréquemment utilisés en théorie du choix social : le critère de majorité (*majority criterion*), le critère de Condorcet (*Condorcet criterion*), et l'indépendance locale des alternatives non pertinentes (*local independance of irrelevant alternatives criterion*). Pour chacun de ces

trois critères, nous regardons à quelles conditions sur les valeurs des vecteurs B et T (voir modèle présenté en Section 4.2) ce critère est respecté.

Pour rappel, $R = [r_1, ..., r_m]$ est une liste de classements, B et T sont deux vecteurs réels de taille 6, $f = S^{(B,T)}$ est une KCF et M_f représente l'ensemble des classements consensuels optimaux de R vis-à-vis de f.

Les critères d'équité ont été initialement définis dans un contexte où les classements d'entrée sont des permutations et où le classement consensuel est unique (ce qui est le cas pour les méthodes Borda, Copeland, Schulze, ...). Or, non seulement une KCF est capable de prendre en entrée des classements incomplets avec égalités, mais en plus on peut avoir plusieurs classements consensuels optimaux. Afin de prendre en compte ces spécificités, nous donnons quelques définitions préliminaires. On dira ainsi que :

- un candidat $x \in U$ est un *vainqueur fort* si pour tout $c \in M_f$ et $y \in U \backslash \{x\}$ on a $x \prec_c y$;
- un candidat $x \in U$ est un *vainqueur possible* s'il existe $c \in M_f$ tel que $x \preceq_c y$, pour tout $y \in U \backslash \{x\}$;
- un candidat $x \in U$ est *préféré* (respectivement *strictement préféré*) par un électeur i si $x \in r_i[1]$ (respectivement si $r_i[1] = \{x\}$).

4.4.3 Critère de majorité

Le critère de majorité stipule que si un candidat est strictement préféré par une stricte majorité d'électeurs, alors ce candidat doit gagner l'élection.

Dans notre contexte, on dira qu'une KCF respecte ce critère si et seulement si lorsqu'un candidat est strictement préféré par une stricte majorité d'électeurs, alors c'est un vainqueur fort de l'élection.

Théorème 4.4.1. *Une KCF $S^{(B,T)}$ respecte le critère de majorité sur $C(U) \times A(U)^{<\infty}$ si et seulement si*

— $B[2] = B[5] - B[4]$ *et*

— $\min\limits_{i \in \{1,4\}} (T[i] - B[i]) \geq \max\limits_{j \in \{2,3,5,6\}} (B[j] - T[j])$.

Démonstration : Nous montrons que les deux conditions sont nécessaires par contraposition dans le cas particulier où $U = \{X, Y\}$; la généralisation à n'importe quel U se fait en remplaçant Y par tous les éléments de $U \setminus \{X\}$ mis à égalité.

Commençons par supposer que $B[2] \neq B[5] - B[4]$. Dans ce cas, selon le Lemme 4.4.4, il existe $R = (r_1, \ldots, r_m) \in A(U)^{<\infty}$ tel que

$$\sum_{i \in \{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}$$

mais $[\{X\}, \{Y\}] \notin M_f$, donc X n'est pas un vainqueur fort de l'élection.

Supposons maintenant que $\min\limits_{i \in \{1,4\}} (T[i] - B[i]) < \max\limits_{j \in \{2,3,5,6\}} (B[j] - T[j])$. Selon le Lemme 4.4.5, il existe

$R \in P(U)^{<\infty}$ tel que

$$\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \sum_{j\in\{2,3,5,6\}} \Omega^R_{X,Y}[j]$$

mais $[\{X\},\{Y\}] \notin M_f$, donc X n'est pas un vainqueur fort de l'élection.

Maintenant, prouvons que les deux conditions sont suffisantes. Supposons que $B[2] = B[5] - B[4]$ et $\min\limits_{i\in\{1,4\}}(T[i] - B[i]) \geq \max\limits_{j\in\{2,3,5,6\}}(B[j] - T[j])$.

En vue d'utiliser le Lemme 4.4.3, on commence par montrer que $\min\limits_{i\in\{1,4\}}(T[i] - B[i]) > 0$, c'est-à-dire $T[1] > B[1]$ et $T[4] > B[4]$. Par définition d'une KCF (Définition 4.2.2), $B[1] = 0$, $B[2] > 0$, $T[1] = T[2]$ et $T[4] = T[5]$. Donc si $T[1] \leq B[1]$ alors $T[2] \leq 0$ d'où $B[2] - T[2] > 0$. Si $T[4] \leq B[4]$, comme $B[5] - B[4] = B[2] > 0$ on a $B[5] > T[4] = T[5]$ d'où $B[5]-T[5] > 0$. Dans les deux cas, $\max\limits_{j\in\{2,3,5,6\}}(B[j]-T[j]) > 0$ d'où $\min\limits_{i\in\{1,4\}}(T[i]-B[i]) > 0$.

Nous devons montrer qu'un candidat $w \in U$ qui est strictement préféré par une stricte majorité de votants est un vainqueur fort de l'élection.

Soit $R = [r_1, ..., r_m]$ tel que w est strictement préféré par m_1 votants et non strictement préféré par m_2 votants avec $m_1 > m_2$.

Soit $c = [P_1, ..., P_k]$ un classement consensuel de R (i.e. un classement complet de U) tel que $P_1 \neq \{w\}$, et soit $i \geq 1$ tel que $w \in P_i$. On va montrer que c n'est pas optimal. Considérons le classement complet c' tel que :

— $c'[1] = \{w\}$, et

— pour chaque $x, y \in U\backslash\{w\}$, $x \leq_c y$ si et seulement si $x \leq_{c'} y$.

On peut noter que la différence entre c et c' est que les éléments qui sont à égalité avec w dans c (autrement dit, $P_i\backslash\{w\}$) sont classés après w dans c', et que les éléments qui sont classés avant w dans c (autrement dit, $V = \bigcup\limits_{j<i} P_j$) sont classés après w dans c'. Soit $A = f(c', R) - f(c, R)$. En utilisant la Définition 4.2.1,

$$A = \sum_{y\in P_i\backslash\{w\}} before(w,y) - tied(w,y) + \sum_{y\in V} before(w,y) - before(y,w).$$

Par le Lemme 4.4.3, on obtient que pour chaque $y \in U\backslash\{w\}$,

$$before(w,y) - tied(w,y) < 0,$$

et par le Lemme 4.4.2, on obtient que

$$before(w,y) - before(y,w) < 0.$$

On a donc $f(c', R) < f(c, R)$, autrement dit, c ne peut pas être un classement consensuel optimal de R vis-à-vis de f. Ainsi $P_1 = \{w\}$ pour tout classement consensuel optimal, c'est-à-dire w est un vainqueur fort.

Corollaire 4.4.1. *Une KCF $S^{(B,T)}$ respecte le critère de majorité sur $C(U) \times C(U)^{<\infty}$ si et seulement si*

$$T[1] \geq \max_{i \in \{2,3\}} (B[i] - T[i]).$$

Démonstration : A nouveau, nous montrons la nécessité par contraposition dans le cas particulier où $U = \{X, Y\}$. Supposons que

$$T[1] < \max_{i \in \{2,3\}} (B[i] - T[i]).$$

On sait que $T[1] = T[1] - B[1]$ puisque $B[1] = 0$ d'après la Définition 4.2.2. Selon le Lemme 4.4.5, il existe $R \in A(U)^{<\infty}$ tel que

- $\Omega^R_{X,Y}[1] > \Omega^R_{X,Y}[2] + \Omega^R_{X,Y}[3]$,
- $\Omega^R_{X,Y}[4] = \Omega^R_{X,Y}[5] = \Omega^R_{X,Y}[6] = 0$ c'est-à-dire $R \in C(U)^{<\infty}$,
- $[\{X\}, \{Y\}]$ n'est pas un classement consensuel optimal de R vis-à-vis de f.

Les deux premiers points impliquent que X est strictement préféré par une stricte majorité d'électeurs, tandis que le dernier implique que X n'est pas un vainqueur fort. En conséquence, le critère de majorité n'est pas satisfait sur $C(U) \times C(U)^{<\infty}$.

Supposons maintenant que $T[1] \geq \max_{i \in \{2,3\}} (B[i] - T[i])$. Soit $\alpha \in \mathbb{R}$ tel que $\alpha > max\{T[1], T[2]\}$ (donc $\alpha > T[2] - B[2]\}$). Soit $f' = S^{(B',T')}$ une KCF telle que

$$B' = (B[1], B[2], B[3], 0, B[2], 0) \text{ et } T' = (T[1], T[2], T[3], \alpha, \alpha, \alpha).$$

Alors $B'[2] = B'[5] - B'[4]$. De plus $\min_{i \in \{1,4\}} (T'[i] - B'[i]) = min\{T[1], \alpha\} = T[1]$ et $\max_{j \in \{2,3,5,6\}} (B'[j] - T'[j]) = max\{B[2] - T[2], B[3] - T[3], B[2] - \alpha, -\alpha\} = max\{B[2] - T[2], B[3] - T[3]\} \leq T[1]$. Donc $\min_{i \in \{1,4\}} (T'[i] - B'[i]) \geq \max_{j \in \{2,3,5,6\}} (B'[j] - T'[j])$.

Par le Théorème 4.4.1, on obtient que f' respecte le critère de majorité sur $C(U) \times A(U)^{<\infty}$, et donc f' respecte le critère de majorité sur $C(U) \times C(U)^{<\infty}$. De plus, par la Proposition 4.2.6, pour chaque $(c, R) \in C(U) \times C(U)^{<\infty}$, on a $f(c, R) = f'(c, R)$. En conséquence, f respecte le critère de majorité sur $C(U) \times C(U)^{<\infty}$.

4.4.4 Critère de Condorcet

Le *critère de Condorcet* stipule que si un candidat w bat chaque autre candidat au sens d'un duel à la majorité stricte (w est alors appelé *vainqueur de Condorcet*), w devrait gagner l'élection. Dans notre contexte, nous stipulons

qu'une KCF respecte ce critère si et seulement si un candidat w est un vainqueur fort de l'élection dès que

$$\Omega^R_{w,y}[1] + \Omega^R_{w,y}[4] > \frac{m}{2} \quad \forall y \neq w \in U.$$

On va montrer que les KCF qui respectent le critère de Condorcet sont exactement les mêmes que celles qui respectent le critère de majorité.

Théorème 4.4.2. *Une KCF $S^{(B,T)}$ respecte le critère de Condorcet sur $C(U) \times A(U)^{<\infty}$ si et seulement si*

— $B[2] = B[5] - B[4]$ *et*

— $\min\limits_{i\in\{1,4\}} (T[i] - B[i]) \geq \max\limits_{j\in\{2,3,5,6\}} (B[j] - T[j])$.

Démonstration : Pour prouver que les conditions sont nécessaires, nous procédons une fois de plus par contraposition dans le cas particulier où $U = \{X, Y\}$. Supposons d'abord que $B[2] \neq B[5] - B[4]$. Par le Lemme 4.4.4, il existe $R = [r_1, \ldots, r_m] \in A(U)^{<\infty}$ une liste de classements telle que $\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}$ mais $[\{X\}, \{Y\}] \notin M_f$ c'est à dire que X n'est pas un vainqueur fort. La première condition est donc nécessaire.

Supposons maintenant que $\min\limits_{i\in\{1,4\}} (T[i] - B[i]) < \max\limits_{j\in\{2,5,6\}} (B[j] - T[j])$. Par le Lemme 4.4.5, il existe $R \in A(U)^{<\infty}$ une liste de classements telle que $\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \sum_{j\in\{2,3,5,6\}} \Omega^R_{X,Y}[j]$ (donc $\sum_{i\in\{1,4\}} \Omega^R_{X,Y}[i] > \frac{m}{2}$), mais $[\{X\}, \{Y\}] \notin M_f$, c'est à dire que X n'est pas un vainqueur fort. La deuxième condition est donc également nécessaire.

Pour prouver que les conditions sont suffisantes, on repend exactement les mêmes étapes que pour la preuve du Theorème 4.4.1 pour montrer que si $B[2] = B[5] - B[4]$ et $\min\limits_{i\in\{1,4\}} (T[i] - B[i]) \geq \max\limits_{j\in\{2,3,5,6\}} (B[j] - T[j])$, alors un candidat $w \in U$ tel que $\Omega^R_{w,y}[1] + \Omega^R_{w,y}[4] > \frac{m}{2}$ pour chaque $y \in U \setminus \{w\}$ est un vainqueur fort.

Corollaire 4.4.2. *Une KCF $S^{(B,T)}$ respecte le critère de Condorcet sur $C(U) \times C(U)^{<\infty}$ si et seulement si*

$$T[1] \geq \max_{i\in\{2,3\}} (B[i] - T[i]).$$

Démonstration :

Nous montrons que la condition est nécessaire par contraposition dans le cas particulier où $U = \{X, Y\}$; la généralisation à n'importe quel U se fait en remplaçant Y par tous les élements de $U \setminus \{X\}$ mis à égalité.

Supposons que $T[1] < \max\limits_{i\in\{2,3\}} (B[i] - T[i])$. On sait que $T[1] = T[1] - B[1]$ puisque $B[1] = 0$ d'après la Définition 4.2.2. Selon le Lemme 4.4.5, il existe $R \in A(U)^{<\infty}$ tel que

— $\Omega^R_{X,Y}[1] > \Omega^R_{X,Y}[2] + \Omega^R_{X,Y}[3]$,

— $\Omega^R_{X,Y}[4] = \Omega^R_{X,Y}[5] = \Omega^R_{X,Y}[6] = 0$, c'est-à-dire $R \in C(U)^{<\infty}$,

— $[\{X\}, \{Y\}]$ n'est pas un classement consensuel optimal de R vis-à-vis de f.

Les deux premiers points impliquent que $\Omega^R_{X,y}[1] + \Omega^R_{X,y}[4] > \frac{m}{2}\ \forall y \neq X \in U$, tandis que le dernier implique que X n'est pas un vainqueur fort. En conséquence, le critère de Condorcet n'est pas satisfait sur $C(U) \times C(U)^{<\infty}$.

Réciproquement, supposons que $T[1] \geq \max\limits_{i \in \{2,3\}} (B[i] - T[i])$. Soit $\alpha \in \mathbb{R}$ tel que $\alpha > max\{T[1], T[2]\}$. Soit $f' = S^{(B',T')}$ une KCF telle que $B' = (B[1], B[2], B[3], 0, B[2], 0)$ et $T' = (T[1], T[2], T[3], \alpha, \alpha, \alpha)$. Alors $B'[2] = B'[5] - B'[4]$. De plus $\min\limits_{i \in \{1,4\}} (T'[i] - B'[i]) = min\{T[1], \alpha\} = T[1]$ et $\max\limits_{j \in \{2,3,5,6\}} (B'[j] - T'[j]) = max\{B[2] - T[2], B[3] - T[3], B[2] - \alpha, -\alpha\} = max\{B[2] - T[2], B[3] - T[3]\} \leq T[1]$. Donc $\min\limits_{i \in \{1,4\}} (T'[i] - B'[i]) \geq \max\limits_{j \in \{2,3,5,6\}} (B'[j] - T'[j])$. Par le Theorème 4.4.2, f' respecte le critère de Condorcet sur $C(U) \times A(U)^{<\infty}$. Donc f' respecte le critère de Condorcet sur $C(U) \times C(U)^{<\infty}$. De plus, par la Propriété 4.2.6, $f(c, R) = f'(c, R)$ $\forall (c, R) \in C(U) \times C(U)^{<\infty}$. On peut conclure que f respecte le critère de Condorcet sur $C(U) \times C(U)^{<\infty}$.

Conséquences pour des scores classiques

En revenant sur la Table 4.2 qui récapitule les valeurs des vecteurs pour les KCFs de la littérature, on peut conclure que le score de Kemeny S^p avec le paramètre de pénalité p défini par Fagin *et al.* [47] respecte le critère de majorité et le critère de Condorcet si et seulement si $p \geq 0.5$.

Par ailleurs, le sore de Kemeny généralisé à la pseudo-distance (Définition 2.3.6) et le score de Kemeny unifiant (Définition 4.2.6) respectent ces deux critères tandis que le score de Kemeny généralisé induit (Définition 4.2.5) ne respecte aucun de ces deux critères.

Remarque. *Le score de Kemeny, lui, respecte bien ces deux critères. En effet, même si dans la Table 4.2 on observe que son coefficient $B[4]$ et son coefficient $B[5]$ sont nuls, ces coefficients ont été choisis arbitrairement (et donc ne sont pas en gras dans la Table 4.2) vu qu'ils ne rentrent jamais en compte dans les calculs étant donné le domaine de définition du score de Kemeny ($P(U) \times P(U)^{<\infty}$). Le score de Kemeny est équivalent (sur son domaine de définition) à toute KCF $S^{(B,T)}$ telle que $B[2] = 1$. En particulier, le score de Kemeny est équivalent (sur son domaine de définition) au score de Kemeny généralisé à la pseudo-distance qui respecte le critère de majorité et le critère de Condorcet.*

Nous allons maintenant discuter du fait que le respect de ces critères n'est pas une fin en soi. En effet, ces critères ne sont pas pertinents dans n'importe quel contexte. Une KCF les respectant et une ne les respectant pas peuvent donc être complémentaires, la première étant alors pertinente sur certains jeux de données (certains contextes) et la deuxième sur d'autres jeux de données (d'autres contextes).

Dans quel cas peut-on vouloir ne pas respecter ces critères ?

S'il paraît à première vue raisonnable de respecter ces critères, dans certaines situations, on va volontairement vouloir les laisser de côté. Reprenons l'exemple des données multi-omiques dans lesquelles certaines protéines n'apparaissent pas dans certains classements d'entrée puisque leur hydrophobie n'a pas permis leur dosage. Nous

avons à faire à un biais expérimental qui empêche la visibilité de certains éléments. Pour rappel, dans cette situation, un élément absent est incomparable avec un élément présent. Ainsi, le fait d'être premier dans une stricte majorité de classements ne suffit pas à garantir la première place dans les classements consensuels optimaux. Il faut prendre en compte, pour chaque paire d'éléments, le nombre de classements dans lesquels les deux éléments sont présents.

De la même manière, lorsque des utilisateurs classent des films par ordre de préférence, il est fort probable que chaque utilisateur n'ait pas vu tous les films à classer. Ainsi, les films non vus par un utilisateurs ne peuvent pas être comparés avec les films vus. Ainsi, si un film a été vu par très peu d'utilisateurs mais a été très bien classé par ceux qui l'ont vu, ce film peut légitimement prétendre à une première place dans le classement consensuel.

Ainsi, il peut être important dans certains contextes de s'abstraire du critère de majorité et du critère de Condorcet, en particulier lorsque l'objectif est de mettre en avant une perle rare.

4.4.5 Généralisation du critère de Condorcet

Une KCF peut respecter ou ne pas respecter le crtitère de Condorcet selon les valeurs des vecteurs de pénalité B et T (voir Theorème 4.4.2). Mais si nous modifions la définition de "majorité d'électeurs", nous obtenons un critère respecté par n'importe quelle KCF.

Propriété 4.4.1. *Soit un candidat $w \in U$. Si $\forall y \in U\backslash\{w\}$, $before(w,y) < before(y,w)$ et $before(w,y) < tied(y,w)$, alors w est un vainqueur fort.*

Démonstration : Soit $c = [P_1, \ldots, P_k]$ un classement complet tel que $P_1 \neq \{w\}$. Soit $i \geq 1$ tel que $w \in P_i$. Soit $c' = [P'_1, \ldots, P'_k]$ le classement complet tel que :

- $P'_1 = \{w\}$, et
- pour chaque paire $x, y \in U\backslash\{w\}$, $x \preceq_c y$ si et seulement si $x \preceq_{c'} y$.

On peut noter que la différence entre c et c' est que les éléments qui sont à égalité avec w dans c, à savoir $P_i\backslash\{w\}$, sont classés après w dans c', et que les éléments qui sont classés avant w dans c, à savoir $V = \bigcup_{j<i} P_j$, sont classés après w dans c'. En posant $A = f(c', R) - f(c, R)$ et en utilisant la Définition 4.2.1, on obtient

$$\begin{aligned} A &= \sum_{y \in P_i\backslash\{w\}} before(w,y) - tied(w,y) \\ &+ \sum_{y \in V} before(w,y) - before(y,w) < 0, \end{aligned}$$

puisque $before(w,y) < before(y,w)$ et $before(w,y) < tied(y,w)$ pour chaque $y \in U\backslash\{w\}$. En conséquence, c ne peut pas être un classement consensuel optimal. On en conclut que pour tout classement consensuel optimal c, $c[1] = \{w\}$, i.e. w est un vainqueur fort.

Dans cette généralisation, on perd le lien entre l'idée de gagner une élection et celle d'être avant les autres candidats dans une stricte majorité de classements. Intuitivement, si on est capable de trouver un candidat w tel que le coût de mettre ce candidat avant n'importe quel autre candidat y est le coût minimal pour la paire (w, y), alors w se retrouve en première position de tous les classements consensuels optimaux.

4.4.6 Indépendance locale des alternatives non pertinentes

En théorie du choix social, le terme *alternative* désigne un élément de l'ensemble U à savoir un candidat dans ce contexte d'élections.

La méthode Kemeny-Young ne respecte pas un des critères les plus cités en théorie du choix social à savoir le critère d'indépendance des alternatives non pertinentes. Young propose alors dans [86] une version "locale" de ce critère, cette fois respectée par la méthode Kemeny-Young.

Une méthode d'agrégation de classements respecte le critère d'indépendance locale des alternatives non pertinentes (LIIA) si les conditions suivantes sont respectées :

- si le candidat $x \in U$ qui est en dernière position du classement consensuel est supprimé des classements d'entrée, alors la position relative des candidats restants dans le nouveau classement consensuel ne change pas (en particulier, le gagnant ne change pas),
- si le candidat $w \in U$ qui est en première position du classement consensuel est supprimé des classements d'entrée, alors la position relative des candidats restants dans le nouveau classement consensuel ne change pas (la position de chaque candidat restant est simplement réduite de 1 dans le nouveau classement consensuel).

Dans notre contexte, on dira qu'une KCF respecte ce critère si, pour toute liste de classements R et toute classement consnesuel optimal $[P_1, \ldots, P_k]$ de R, les deux conditions suivantes sont vérifiées :

- si les éléments dans P_k sont supprimés , alors $[P_1, \ldots, P_{k-1}]$ est un classement consensuel optimal pour la nouvelle liste de classements
- si les éléments dans P_1 sont supprimés, alors $[P_2, ..., P_k]$ est un classement consensuel optimal pour la nouvelle liste de classements.

Théorème 4.4.3. *Pour tous $B, T \in \mathbb{R}^6$, $S^{(B,T)}$ respecte le critère d'indépendance locale des alternatives non pertinentes (LIIA) sur $C(U) \times A(U)^{<\infty}$.*

Démonstration : Soit $f = S^{(B,T)}$, $c = [P_1, ..., P_k]$ un classement consensuel optimal de R vis-à-vis de f, $U' = U \backslash P_1$, R' la projection de R sur U' et $c_2 = [P'_1, ..., P'_{k'}] \in C(U')$.

Soit $c' = [P_1, P'_1, ..., P'_{k'}]$ un classement consensuel de R (c' est complet). Posons $A = f(c, R) - f(c', R)$,

alors $A \leq 0$ puisque c est un classement consensuel optimal pour R (Définition 4.2.3). Nous savons que

$$A = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} before(x,y) + \frac{1}{2} \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} tied(x,y)$$

$$- \sum_{\substack{x \neq y \in U \\ x \prec_{c'} y}} before(x,y) - \frac{1}{2} \sum_{\substack{x \neq y \in U \\ x \equiv_{c'} y}} tied(x,y)$$

Comme pour chaque $x, y \in P_1$, $x \equiv_c y$ et $x \equiv_{c'} y$, nous pouvons ignorer les paires d'éléments de P_1 dans le calcul de A :

$$A = \sum_{\substack{x \neq y \in U' \\ x \prec_c y}} before(x,y) + \frac{1}{2} \sum_{\substack{x \neq y \in U' \\ x \equiv_c y}} tied(x,y) - \sum_{\substack{x \neq y \in U' \\ x \prec_{c'} y}} before(x,y) - \frac{1}{2} \sum_{\substack{x \neq y \in U' \\ x \equiv_{c'} y}} tied(x,y)$$

$$= f([P_2, \ldots, P_k], R') - f(c_2, R')$$

Comme $A \leq 0$, on peut conclure que $f([P_2, ..., P_k], R') \leq f(c_2, R')$ pour tout $c_2 \in C(U')$. Donc $[P_2, \ldots, P_k]$ est un classement consensuel optimal pour R'.

Par un raisonnement analogue, $[P_1, \ldots, P_{k-1}]$ est un classement consensuel optimal pour la projection R' de R sur $U' = U \setminus P_k$.

Utilisation du LIIA.

Le critère LIIA garantit qu'après le calcul d'un classement consensuel optimal, les premiers éléments (respectivement derniers éléments) conservent leur ordre relatif si les éléments les moins intéressants (respectivement les plus intéressants) sont supprimés des classements d'entrée.

Ce critère est intéressant dans certaines situations concrètes, par exemple pour envisager des expériences biologiques dans le but d'étudier l'importance de certains gènes liés à une maladie donnée. Si les chercheurs qui mènent les expériences ne veulent pas considérer les gènes du premier *bucket* du classement consensuel optimal (qui seraient déjà très étudiés), ils peuvent directement considérer les gènes du deuxième *bucket*. Il n'y a pas besoin de retirer des classements d'entrée les gènes de ce premier *bucket* et de relancer le calcul. On obtient le même résultat si les chercheurs veulent supprimer les derniers gènes du classement consensuel (dans l'hypothèse que leur présence semble non pertinente).

Par ailleurs, le LIIA est un critère très utile lors de la conception d'algorithmes. En effet, supposons que pour un R donné une condition suffisante nous garantisse que dans au moins un classement consensuel optimal, l'élément x est seul en première position (le critère de Condorcet est une condition suffisante qui va dans ce sens). Grâce au LIIA, le problème initial se divise en deux sous-problèmes dont le premier consiste à trier l'élément x seul et le deuxième consiste à trier les éléments restants. Pour être plus exact, il faut que la condition suffisante per-

mette d'isoler l'ensemble des éléments des k premiers (ou derniers) *buckets* d'au moins un classement consensuel optimal. Ainsi, le LIIA est très corrélé à l'algorithme de partitionnement ParCons décrit dans le Chapitre 3.

4.5 CoRankCo : plateforme pour l'agrégation de classements

L'ensemble des concepts présentés dans ce chapitre a été implémenté dans une plateforme en ligne que nous présentons maintenant. L'objectif de cette plateforme est de permettre le calcul de classements consensuels pour des jeux de données incomplets avec égalités, avec la possibilité de choisir sa KCF à travers les vecteurs de pénalité.

4.5.1 Présentation du logiciel

Nous présentons ici la plateforme CoRankCo (*Consensus Ranking Computation*) développée pendant la thèse. La partie algorithmique représente 3126 lignes de code python pour 19 classes. L'interface utilisateur et la gestion des bases de données ont été gérées par Bryan Brancotte. CoRankCo a été implémenté en python 3.6 et utilise le framework Django offrant ainsi une interface Web aux utilisateurs, accessible sur `https://corankco.lri.fr`. Tous les jeux de données, les résultats des exécutions et les informations utilisateur sont stockés dans une base de données présentée en Figure 4.1.

4.5.2 Utilisation du logiciel

La Figure 4.2 présente l'interface utilisateur de CoRankCo. Celle-ci peut être présentée en trois parties distinctes détaillées ci-dessous.

Les jeux de données

Les utilisateurs peuvent utiliser les jeux de données présents dans la base (s'ils sont publics c'est-à-dire visibles et utilisables par tous les utilisateurs), saisir leurs jeux de données dans la base s'ils ont un compte (en choisissant si le jeu de données est public ou non), ou bien rentrer manuellement les classements à agréger dans un champ texte. Dans tous les cas, un jeu de données doit être présenté comme décrit ci-dessous.

- Les classements sont séparés par un passage à la ligne. Chaque classement peut être étiqueté par un nom suivi de " :=".
- Après le " :=" si le classement est étiqueté, ou en début de ligne sinon, un premier "[" signifie le début du classement .
- Chaque *bucket* commence par un "[" et se termine par un "]". Au sein d'un *bucket*, les éléments sont séparés par une virgule. Un *bucket* ne peut pas être vide.

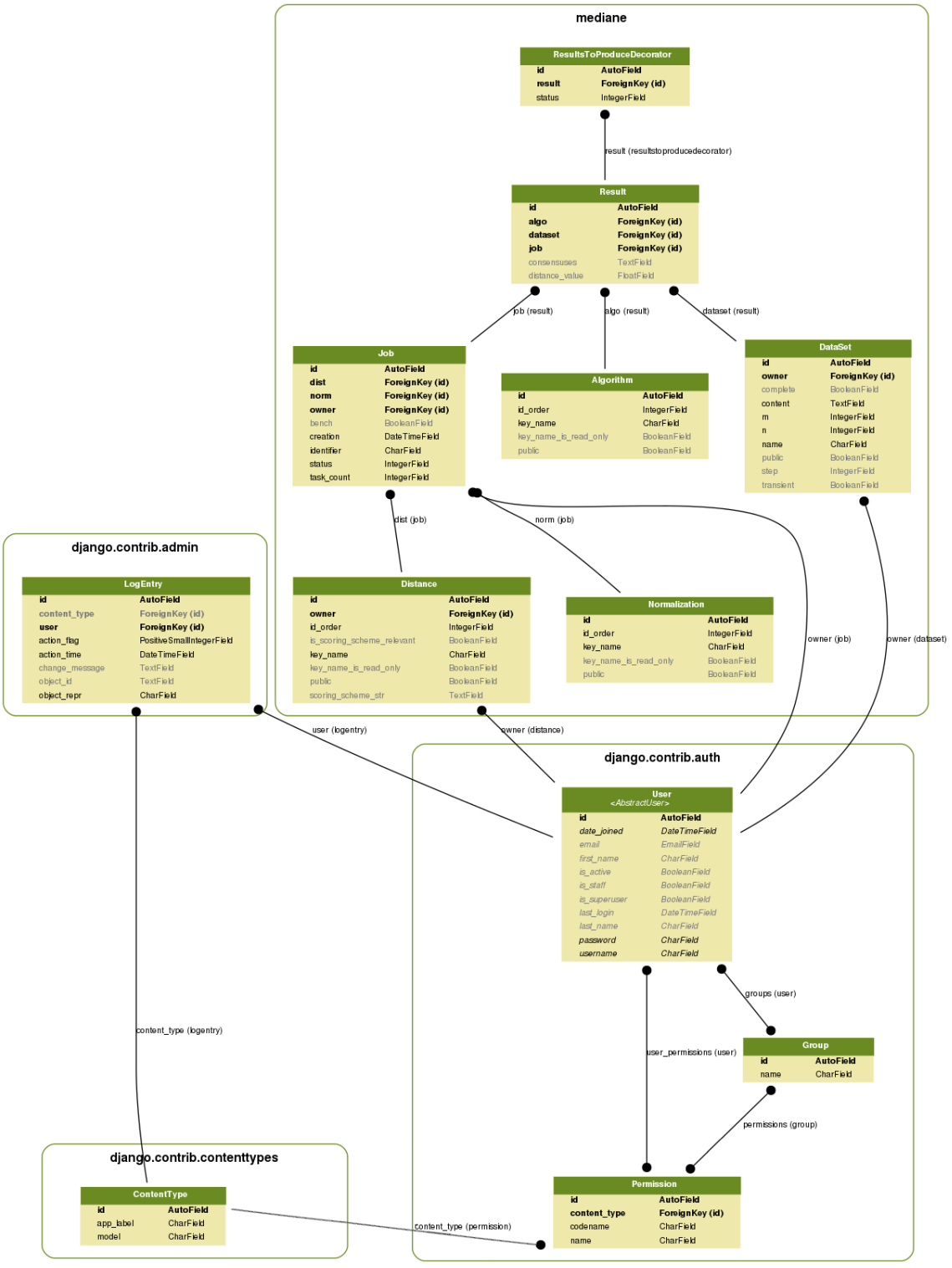


FIGURE 4.1 – Base de données de l'outil CoRankCo

- Les *buckets* sont séparées par une virgule.
- Un "]" avant un passage à la ligne indique la fin du classement.

Par exemple,

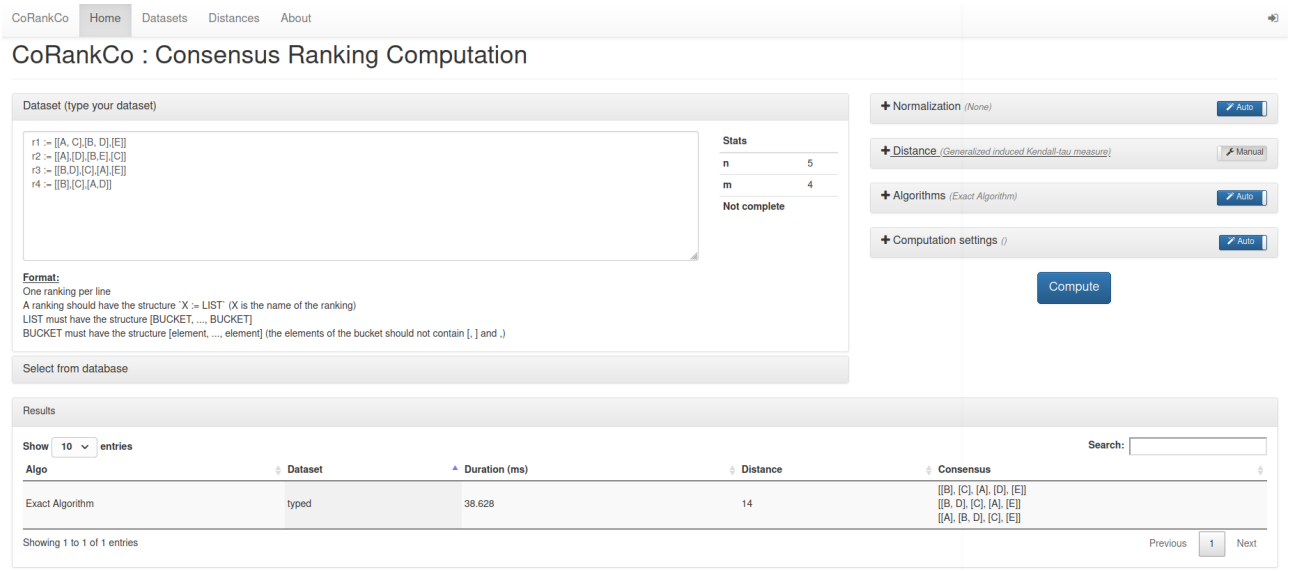


FIGURE 4.2 – Interface utilisateur de l'outil CoRankCo

r1 := [[A, B], [C], [E]]

r2 := [[D, B], [A]]

correspond à un jeu de données comportant deux classements. Le logiciel affiche automatiquement trois caractéristiques du jeu de données écrit manuellement : le nombre d'éléments, le nombre de classements, et complet ou incomplet.

Il est également possible de choisir des jeux de données contenus dans la base de données. La plateforme contient ainsi les jeux de données utilisés lors des expériences présentées dans le Chapitre 4.6, ainsi que des jeux de données de votes de la base de données publique *preflib* `https://www.preflib.org/`. Les parseurs permettant de saisir les jeux de données dans un format adapté ont été implémentés par Kenyu Kobayashi pendant un stage de troisième année de licence.

Les paramètres pour le calcul de classement consensuel

La seconde partie, permettant de paramétrer le classement consensuel, contient différents panels.

- Panel de *normalisation* : Ce premier panel permet de sélectionner la normalisation du jeu de données par projection ou unification. Nous rappelons que la projection (Définition 2.3.1) consiste à retirer des classements les éléments qui ne sont pas présents dans chaque classement et que l'unification (Définition 2.3.2) consiste à ajouter à la fin de chaque classement incomplet un *bucket* d'unification contenant l'ensemble des éléments manquants.
- Panel *distance* : Ce panel permet de choisir le score de Kemeny que l'on souhaite considérer pour le calcul

de classement consensuel. Si un algorithme choisi (voir ci-dessous) n'est pas compatible avec le score de Kemeny considéré, le calcul de classement consensuel n'a pas lieu.

Les adaptations les plus courantes du score de Kemeny aux classements incomplets sont accessibles à tous les utilisateurs et les coefficients des vecteurs B et T associés sont rappelés. Si un utilisateur veut utiliser des coefficients différents, il lui faut alors se créer un compte et ajouter la distance voulue dans la base en indiquant les coefficients des vecteurs de pénalité B et T.

- Panel *algorithmes* : Ce panel permet de sélectionner le ou les algorithmes qui vont être utilisés pour calculer le classement consensuel. Sont aujourd'hui disponibles les douze algorithmes suivants : BioCo, BioConsert, BordaCount, ParCons, CopelandMethod, ExactAlgorithm, ExactAlgorithmPreprocessing, KwikSort, MEDRank, Pick-a-Perm, Repeat Choice, SchulzeMethod. Les algorithmes BioCo, BioConsert, BordaCount, CopelandMethod, KwikSort, MedRank, Pick-A-Perm et Repeat Choice sont décrits dans [25]. Les adaptations pour les données manquantes sont décrites dans la Section 4.3.
- Panel *computation settings* : Ce dernier panel permet d'ajouter des paramètres liés à l'exécution de l'algorithme. Il est par exemple possible de demander une mesure plus précise du temps d'exécution des algorithmes. Ainsi, pour un algorithme et un jeu de données, le calcul est relancé autant de fois que nécessaire pour que le temps d'exécution atteigne deux secondes. Le temps affiché est alors le temps moyen de calcul.

Remarque. *L'algorithme ExactAlgorithm correspond à l'algorithme exact décrit dans la Sous-section 4.3.1. L'implémentation a été faire de sorte que l'algorithme renvoie l'ensemble des classements consensuels optimaux. Le temps de calcul peut être long si le nombre d'éléments dépasse quelques dizaines. A contrario, l'algorithme ExactAlgorithmPreprocessing utilise la décomposition en composantes fortement connexes décrite dans la Section 3.5. Cette version de l'algorithme exact renvoie ainsi un unique classement consensuel optimal. Il est généralement beaucoup plus rapide que le précédent et peut dans certains cas être utilisé sur des jeux de données contenant plus d'une centaine d'éléments. Il est difficile d'estimer le temps de calcul de ce dernier qui peut être très variable pour un même nombre d'éléments à classer. En effet, le temps de calcul est très dépendant de la taille des composantes fortement connexes calculées par l'algorithme ParCons.*

Résultats du calcul de classement(s) consensuel(s)

Pour chaque jeu de données sélectionné et chaque algorithme choisi, le(s) classement(s) consensuel(s) sont calculés et affichés. Certains algorithmes ont la possibilité de renvoyer plusieurs classements consensuels, tous équivalents en terme de score de Kemeny. Pour chaque algorithme et chaque jeu de données, le score de Kemeny associé aux classements consensuels calculés et le temps d'exécution de l'algorithme sont affichés.

Illustration 4.5.1. *Sur la Figure 4.2, on peut voir que l'algorithme exact a renvoyé une liste de trois classements consensuels optimaux :* $[[B],[C],[A],[D],[E]]$, $[[B,D],[C],[A],[E]]$ *et* $[[A],[B,D],[C],[E]]$. *Le score associé à chacun*

de ces classements consensuels est de 14.

4.5.3 Un package python pour l'agrégation de classements incomplets avec égalités

En plus de l'outil en ligne CoRankCo avec interface graphique, nous avons implémenté et mis à disposition de la communauté le *package* python *CoRankCo* présenté à l'adresse suivante : `https://pypi.org/project/corankco/`. Ce package peut être installé sur n'importe quel système d'exploitation. Sur Linux, son installation passe par l'exécution de la commande *pip3 install –user corankco* dans le terminal. Comme pour la version *web*, ce package python permet d'agréger des classements incomplets et avec égalité. Les vecteurs B et T du modèle présenté dans la Section 4.2 peuvent être choisis librement. Ce *package* est en fait une version de CoRankCo sans base de données et ne nécessitant pas de connexion internet. Son objectif est de permettre aux développeurs de faire tourner localement les algorithmes sans devoir gérer une base de données ou une interface graphique. Il est à noter que certaines informations sont disponibles en plus de celles affichées sur la version *web* : une heuristique comme ParCons peut par exemple avertir l'utilisateur que le classement consensuel renvoyé est nécessairement optimal si aucune heuristique n'a été utilisée, et afficher le découpage en composantes fortement connexe.

4.6 Évaluation du modèle : influence du choix de la KCF

Cette section s'intéresse à l'évaluation du modèle paramétré présenté dans ce chapitre. Pour démontrer la pertinence de notre démarche dans l'explicitation et la formalisation de notre modèle, nous mettons en évidence l'impact des choix de valeurs des paramètres sur les résultats obtenus au niveau des classements consensuels. Nous nous intéresserons ensuite à l'influence des paramètres sur la capacité à partitionner le problème initial en sous-problèmes par l'algorithme ParCons introduit au Chapitre 3.

4.6.1 Jeux de données considérés

Pour les expériences de cette section, nous considérons le *Dataset 1* et le *Dataset 2* définis dans les sections 3.8 et 3.9 ainsi que deux *Datasets* supplémentaires que nous décrivons ci-dessous.

Dataset 3 (*Dataset* notes d'étudiants à l'université)**.** *Le Dataset 3 correspond à un ensemble de jeux de données synthétiques que nous avons générés et qui sont inspirés de jeux de données réels. Nous considérons les notes de promotions d'étudiants universitaires. Chaque promotion est formée de deux parcours d'étudiants distincts (i) 280 étudiants qui suivent 14 unités d'enseignement à choisir parmi 17 unités proposées et (ii) 20 étudiants qui suivent 9 unités d'enseignement à choisir parmi ces mêmes 17 unités.*

Pour chaque étudiant, les unités suivies parmi les 17 ont été choisies uniformément au hasard. Les notes (valeurs comprises entre 0 et 20 comme dans le système français) ont été obtenues suivant une loi normale :

$\mathcal{N}(\mu_{standard}, \sigma^2_{standard})$ *pour les 280 étudiants avec* $\mu_{standard} = 10$ *et* $\sigma^2_{standard} = 25$.

$\mathcal{N}(\mu_{special}, \sigma^2_{special})$ *pour les 20 étudiants avec* $\mu_{special} = 16$ *et* $\sigma^2_{special} = 16$.

Si une note générée dépasse 20, nous la remettons à 20 et si elle est inférieure à 0 nous la remettons à 0.

On peut observer que les étudiants issus du deuxième parcours réussissent particulièrement bien leurs examens : la moyenne des étudiants de ce groupe a une forte probabilité de se trouver aux alentours de 16 alors que la moyenne des étudiants du premier groupe a une forte probabilité de se trouver aux alentours de 10.

Nous avons ainsi généré les notes de 1 000 promotions de 300 étudiants (280 pour le groupe 1 et 20 pour le groupe 2). Chacun des 1000 jeux de données obtenus est composé de 17 classements d'étudiants : pour chaque unité d'enseignement, les étudiants l'ayant suivie ont été classés par ordre décroissant de note.

Dataset 4 (*Dataset* classements synthétiques)**.** *Le Dataset 4 est un ensemble de 1000 jeux de données purement synthétiques. Chaque classement de ce jeu de données a été généré en simulant une chaîne de Markov dont les états sont des classements et des transitions permettent de passer d'un classement* $r = [G_1, G_2, ..., G_g]$ *(où les* G_i *désignent les buckets) à un autre en déplaçant un élément. Les règles pour les transitions sont décrites ci-dessous.*

Nous prenons $x \in U$*. Deux cas sont possibles :*

- *Supposons d'abord que* $x \notin dom(r)$*. Alors, avec une probabilité de 0,2 nous ajoutons un nouveau bucket contenant* x *au début de* r*.*
- *Supposons maintenant que* $x \in dom(r)$*. Notons* G_i *le bucket contenant* x*. L'élément* x *peut être :*
  - *décalé de* G_i *vers un nouveau bucket entre* G_i *et* G_{i-1} *sauf si* $|G_i| = 1$
  - *passé de* G_i *à un nouveau bucket entre* G_i *et* G_{i+1} *à moins que* $|G_i| \leq 2$
  - *décalé de* G_i *à* G_{i-1} *à moins que* $i = 1$
  - *décalé de* G_i *à* G_{i+1} *à moins que* $i = g$ *ou* $|G_i| = |G_{i+1}| = 1$
  - *supprimé de* r

  Chaque mouvement a la même probabilité de 0,2 de se produire.

Cette chaîne de Markov a été conçue pour être irréductible *et* apériodique*, ce qui implique qu'elle converge vers une distribution limite unique lorsque le nombre d'étapes tend vers l'infini. De plus, puisque la probabilité de tout mouvement est égale à la probabilité du mouvement inverse, cette distribution limite est uniforme, c'est-à-dire que chaque classement a exactement la même probabilité que les autres.*

Considérons maintenant un ensemble de k *classements identiques, et simulons la chaîne de Markov simultanément sur chacun d'eux, pendant un nombre fixe d'étapes. Plus le nombre d'étapes est élevé, plus les différences*

entre les classements de départ vont croître. Si k est très grand, l'ensemble des classements sera proche d'un ensemble généré uniformément. Ainsi ce procédé permet de générer des listes de classements où la dissimilarité peut varier en fonction du nombre d'étapes. Plus exactement, plus on fait de pas, plus l'indépendance grandit pour les classements vis-à-vis du classement initial.

Nous avons d'abord créé 1000 jeux de données identiques où une unique permutation de 300 éléments est répétée 17 fois. Ensuite, nous avons modifié les jeux de données en utilisant le processus décrit ci-dessus. Nous avons effectué un total de 3000 mouvements dans chaque jeux de données par pas de 300 mouvements.

4.6.2 Interprétation des éléments manquants à travers la différence B[5] - B[4]

Dans cette première série d'expériences, nous faisons varier la valeur de $B[5] - B[4]$ et étudions l'impact de cette différence sur le classement consensuel renvoyé. Nous considérons ici le *Dataset 1* (données biologiques, reformulations de maladies) et le *Dataset 3* (notes d'étudiants à l'université).

Les coefficients $B[4]$ et $B[5]$ sont des paramètres particulièrement intéressants dans une KCF car ils sont directement liés à l'interprétation des données manquantes. Plus précisément $B[4]$ (respectivement $B[5]$) correspond au le coût de placer x avant y dans le classement consensuel pour chaque classement d'entrée tel que x est présent alors que y est absent (respectivement y est présent alors que x est absent).

En conséquence, lorsque les éléments manquants doivent être considérés comme moins importants que les éléments présents dans les classements d'entrée, il est nécessaire de mettre $B[4] < B[5]$. Au contraire, lorsque les éléments manquants ne peuvent pas être considérés comme moins importants que les éléments présents (par exemple non comparables), alors il faut mettre $B[5] = B[4]$. La différence $B[5] - B[4]$ indique à quel point les éléments manquants doivent être pénalisés vis-à-vis des éléments présents.

Supposons d'abord que $B[5] - B[4] = 1$. Dans cette situation, avoir x avant y dans le classement consensuel alors que y est présent et que x est absent dans un classement d'entrée a un coût identique à celui d'une inversion.

Avec un raisonnement analogue, une différence $B[5] - B[4] > 1$ peut être vue comme une sur-pénalisation des éléments manquants vis-à-vis des éléments présents et une différence $0 < B[5] - B[4] < 1$ peut être vue comme une sous-pénalisation des éléments manquants vis-à-vis des éléments présents.

Notez qu'une valeur $B[4] > B[5]$ reste possible mais semble "exotique" si l'on veut considérer des jeux de données réels. De telles valeurs ne seront pas considérées ici.

Dans cette série d'expériences, nous considérons les Datasets 1 et 3. Nous faisons varier la différence $B[5]-B[4]$ en fixant la valeur de $B[4]$ à 0 tout en changeant la valeur de $B[5]$ de 0 à 6. Nous posons également $T[4] =$

$T[5] = B[4]$ pour éviter la création d'égalités artificielles dans les classements consensuels. Les coefficients fixes et variables de (B, T) de cette expérience sont rappelés dans la Table 4.3.

B	0	1	1	0	**B[5]**	0
T	1	1	0	**B[5]**	**B[5]**	0

TABLE 4.3 – Vecteurs de pénalité considérés pour les expériences (i), (ii) et (iii).

Remarque. *Les expériences ci-dessous ont également été effectuées pour des valeurs différentes de* $B[4]$*. Nous avons observé que pour deux paires de paramètres* $(B[4]_1, B[5]_1)$ *et* $(B[4]_2, B[5]_2)$ *telles que* $B[5]_1 - B[4]_1 = B[5]_2 - B[4]_2$*, les résultats étaient rigoureusement identiques. Ceci s'explique en grande partie grâce au Lemme 4.4.1 : pour une liste de classements* R *fixée et deux éléments* $x \neq y$ *de* U*, la différence de coût entre le fait de placer* x *avant* y *et le fait de placer* y *avant* x *dans le classement consensuel ne dépend que de* $B[2]$ *et de la différence* $B[5] - B[4]$*.*

Expérience (i) : lorsque les absences doivent être pénalisées.

Pour cette expérience, on se focalise sur le *Dataset 1* correspondant aux données biologiques de reformulations de maladies.

Pourquoi les éléments manquants doivent être pénalisés ?

Comme les classements sont fournis par une même base de données biologiques (la base de données *Gene* du NCBI), il est logique de considérer que si un gène n'est pas retourné, c'est qu'il est moins pertinent vis-à-vis de la maladie considérée que les gènes retournés. En conséquence, une KCF pertinente doit avoir $B[4] < B[5]$.

Résultat attendu. Nous nous attendons dans cette expérience à ce que poser $B[4] = B[5]$ induise un classement consensuel d'une moins bonne qualité que si l'on pose $B[4] < B[5]$.

Évaluation de la qualité des classements consensuels. Comme dans la Section 3.9, l'évaluation de la qualité des classements consensuels dans ce contexte utilise les *gold standards* fournis par la base de données Orphanet. On va donc comparer le nombre de gènes du *gold standard* présents dans le top-20 de la base de données *Gene* du NCBI avec le nombre de gènes du *gold standard* dans le top-20 des classements consensuels obtenus par les différentes KCF dont la différence $B[5] - B[4]$ varie. Plus précisément, on dit que ConQuR-BioV2 bat le NCBI (respectivement NCBI bat ConQuR-BioV2) si la KCF considérée aboutit à un classement consensuel dont le top-20 contient strictement plus (respectivement strictement moins) de gènes du *gold standard* que le NCBI.

Résultats de l'expérience.

Les résultats sont présentés en Figure 4.3.

De façon intéressante, lorsque $B[5] = B[4]$, ConQuR-BioV2 bat le NCBI sur 34 jeux de données seulement alors que le NCBI bat ConQuR-BioV2 sur 77 jeux de données. Cela signifie que la qualité des classements consensuels fournis par ConQuR-BioV2 est médiocre puisque l'utilisation des synonymes a abouti à une diminution de la qualité

du top-20 par rapport au NCBI qui n'utilise pas les synonymes. En comparaison, lorsque $B[5] - B[4] = 1$, ConQuR-BioV2 bat le NCBI dans 55 jeux de données alors que le NCBI bat ConQuR-BioV2 dans seulement 8 jeux de données. De plus, si la valeur de $B[5] - B[4]$ est trop faible (c'est-à-dire, dans cette expérience, $B[5] - B[4] = 0.1$), alors ConQuR-BioV2 bat le NCBI mais moins significativement. Enfin, les résultats varient très peu une fois qu'un seuil a été atteint (0.3 dans notre expérience).

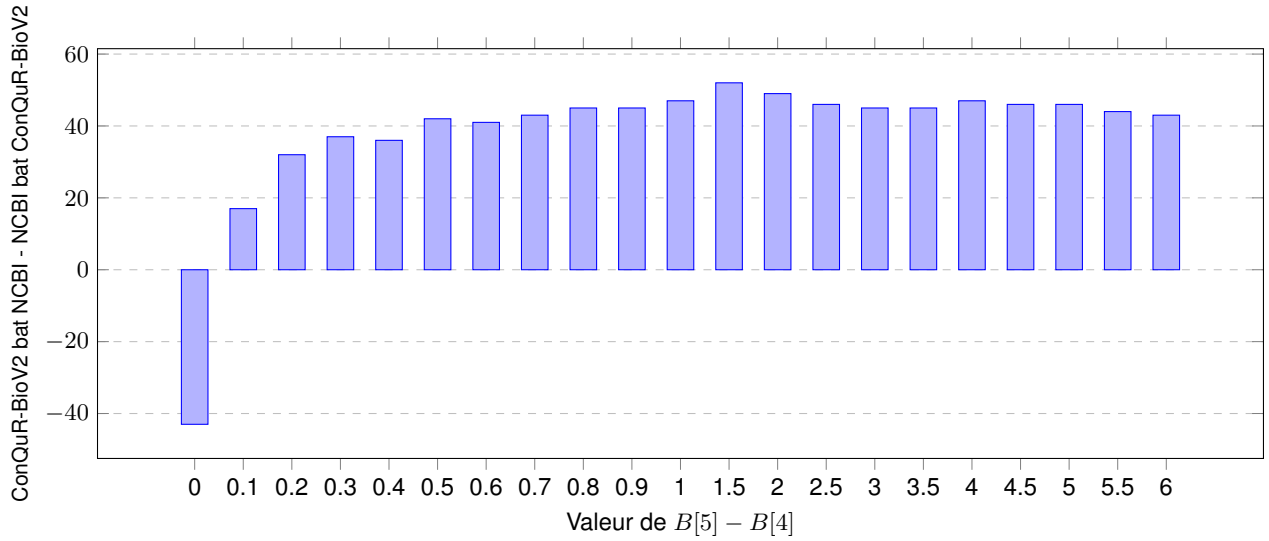


FIGURE 4.3 – Nombre de jeux de données du Dataset 2 tels que ConQuR-BioV2 bat le NCBI moins le nombre de jeux de données tels que le NCBI bat ConQuR-BioV2 en fonction de la valeur de $B[5] - B[4]$.

Conclusions de l'expérience. Les conclusions de cette expérience peuvent être résumées de la façon suivante.

Tout d'abord, comme prévu dans ce contexte où les éléments manquants doivent être pénalisés vis-à-vis des éléments présents, le choix d'une KCF qui ne va pas dans ce sens entraîne le calcul de classements consensuels de mauvaise qualité. Deuxièmement, l'idéal est de faire des choix francs pour éviter le risque de sous-pénaliser les éléments manquants. Enfin, il n'est pas nécessaire de choisir des coefficients "exotiques" : des multiples de 0.5 semblent suffisants pour rendre compte de la nécessité de pénaliser les éléments manquants vis-à-vis des éléments présents.

Expérience (ii) : quand les éléments manquants ne doivent pas être pénalisés vis-à-vis des éléments présents.

Cette expérience se focalise sur le *Dataset 3*. Ce Dataset correspond aux étudiants séparés en deux groupes : un groupe de 280 étudiants qui suivent 14 unités d'enseignement parmi les 17 proposées et un groupe de 20 étudiants qui ne suivent que 9 unités d'enseignements parmi 17. Pour rappel, les étudiants du deuxième groupe on des notes généralement supérieures aux étudiants du premier groupe.

Pourquoi les éléments manquants ne devraient pas être pénalisées vis-à-vis des éléments présents ?

Ici, le contexte est très différent du précédent. Les étudiants qui ont les meilleures notes sont les étudiants inscrits dans un plus petit nombre d'UE. Dans ce contexte, si un étudiant n'est pas inscrit dans une UE, il n'est pas comparable aux étudiants inscrits car il n'est pas évalué pour cette UE. Ainsi, les éléments manquants du classement ne doivent pas être pénalisés vis-à-vis des éléments présents.

Résultat attendu. Ici, les attentes sont à l'opposé de l'expérience précédente : $B[5] = B[4]$ devrait fournir un meilleur classement consensuel que $B[5] > B[4]$.

Évaluation de la qualité des classements consensuels.

La qualité du classement consensuel obtenu est évaluée à l'aide du classement des étudiants selon leur moyenne générale : nous comptons le nombre d'étudiants dans le top-20 du classement consensuel qui sont également dans le top-20 du classement obtenu en considérant leur moyenne générale.

Résultats de l'expérience. Les résultats sont fournis en Figure 4.4.

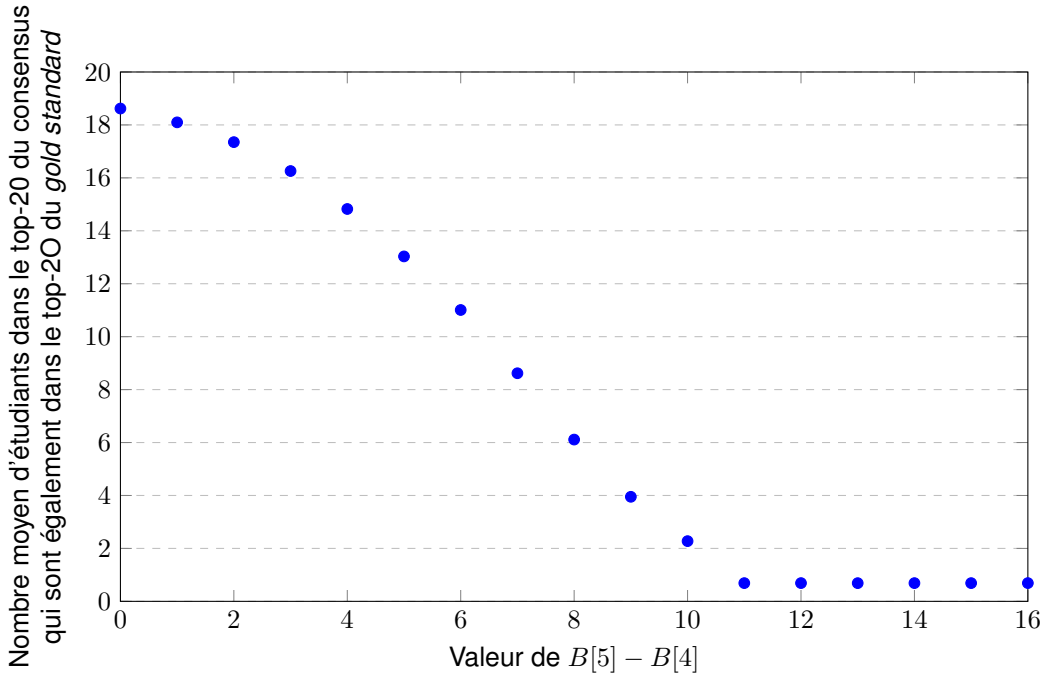


FIGURE 4.4 – Nombre moyen (sur les jeux de données du Dataset 3) d'étudiants dans le top-20 du classement consensuel qui sont également dans le top-20 du classement obtenu en considérant la moyenne générale en fonction de la valeur $B[5] - B[4]$.

Lorsque $B[5] = B[4]$, c'est-à-dire que les éléments absents d'une UE ne sont pas pénalisés, près de 19 étudiants en moyenne sont à la fois dans le top-20 du classement consensuel et dans le top-20 du classement obtenu en considérant la moyenne générale des étudiants. Ce nombre diminue significativement avec l'augmentation de $B[5] - B[4]$ jusqu'à une valeur inférieure à 1 lorsque $B[5] - B[4] > 1$.

Conclusions de l'expérience. A nouveau, les résultats obtenus sont conformes aux attentes. Pénaliser les éléments manquants dans un contexte où cela n'a pas de sens se traduit par le calcul d'un classement consensuel de très mauvaise qualité alors que mettre des coefficients pertinents concernant le contexte conduit au calcul d'un

classement consensuel de bonne qualité.

Les expériences (i) et (ii) ont bien mis en évidence l'importance cruciale de considérer le contexte lorsque l'on veut agréger des classements incomplets, afin de comprendre l'origine des données manquantes et de choisir une adaptation pertinente du score de Kemeny.

Expérience (iii) : quantifier la dissimilarité des classements consensuels.

La troisième expérience utilise le *Dataset 4* à savoir les jeux de données synthétiques de plus en plus dissimilaires obtenus à l'aide des chaînes de Markov.

Objectif de l'expérience.

Dans cette nouvelle expérience, nous utilisons le *Dataset 4* afin d'évaluer la différence entre les classements consensuels obtenus en posant $B[5] - B[4] = 1$ (ce qui était pertinent dans l'expérience (i) mais pas dans l'expérience (ii)) et $B[5] - B[4] = 0$ (ce qui était non pertinent dans l'expérience (ii) mais pas dans l'expérience (i)). Rappelons que dans ce Dataset, les classements d'entrée sont initialement tous identiques. Nous avons effectué un nombre croissant de pas dans la chaîne de Markov détaillée dans la Sous-section 4.6.1 dans le but d'augmenter la dissimilarité. L'objectif est ici d'étudier l'impact du nombre de pas dans la chaîne de Markov sur la dissimilarité des classements consensuels obtenus.

Résultat attendu. On s'attend à ce que la dissimilarité des classements consensuels obtenus avec $B[5]-B[4] = 1$ et $B[5] - B[4] = 0$ augmente avec le nombre de pas dans la chaîne de Markov. En effet, si les classements sont très similaires (peu de pas), le classement consensuel ne devrait pas beaucoup varier. Au contraire, une augmentation du nombre de pas devrait augmenter la dissimilarité des classements d'entrée et donc induire davantage de divergences entre les deux classements consensuels obtenus.

Évaluation de la dissimilarité entre deux classements consensuels. La dissimilarité entre deux classements consensuels est évaluée en divisant le nombre de paires d'éléments de U telles que leur ordre relatif est différent entre les deux classements consensuels par le nombre total de paires d'éléments U.

Résultats de l'expérience. Les résultats de cette expérience sont fournis en Figure 4.5.

Nous pouvons observer que lorsque le nombre de pas est petit (< 300 soit exactement $|U|$), le classement consensuel obtenu est identique entre les deux KCFs. Pour 2 100 pas, à savoir $7 * |U|$, une moyenne de 33 % de paires ne sont pas dans le même ordre relatif entre les deux classements consensuels.

Conclusion de l'expérience. Un nombre même peu élevé de pas dans la chaîne de Markov peuvent induire des différences significatives entre les classements consensuels obtenus par les KCFs considérant $B[5] - B[4] = 1$ et $B[5] - B[4] = 0$.

Conclusion des expériences (i), (ii) et (iii).

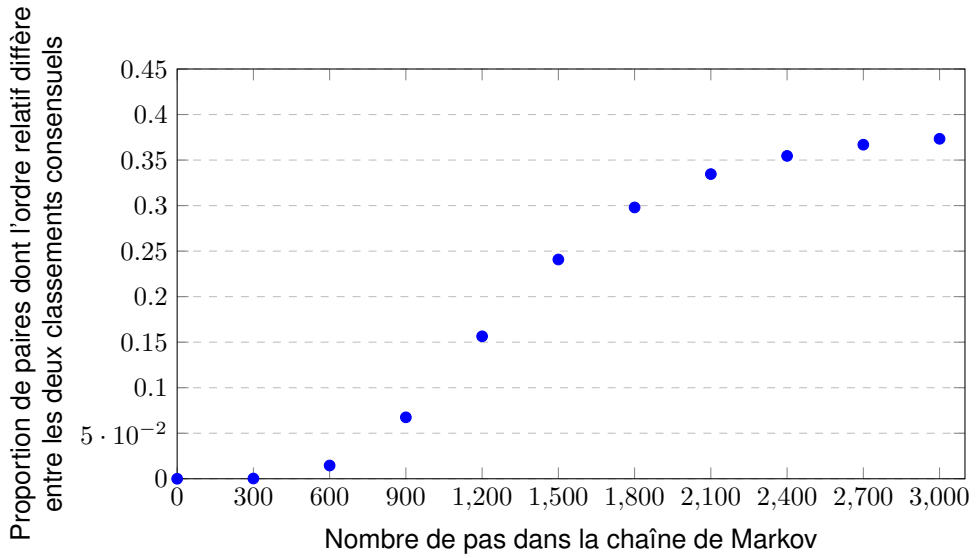


FIGURE 4.5 – Proportion moyenne (sur les jeux de données du Dataset 4) de paires dont l'ordre relatif est différent entre les deux classements consensuels obtenus à partir de deux KCFs différentes en fonction du nombre de pas dans la chaîne de Markov.

Cette série d'expériences met en avant l'impact des coefficients $B[5]$ et $B[4]$ sur les classements consensuels calculés lorsque les classements d'entrée sont incomplets. En particulier, ces expériences montrent que sur des jeux de données réels comportant des classements d'entrée incomplets, il est primordial de comprendre les raisons pour lesquelles certains éléments sont absents de certains classements. Selon que l'absence d'un élément puisse être interprétée comme une "non pertinence de l'élément vis-à-vis du classement" (gène non associé à une reformulation de maladie) ou d'une "incapacité à comparer l'élément avec d'autres éléments" (étudiants non inscrits à une unité d'enseignement), la valeur de $B[5] - B[4]$ a un fort impact sur le classement consensuel calculé. Elle doit être positive dans le premier cas ($B[4]$ à 0 et $B[5]$ à 1 semblent être pertinents) et nulle dans le deuxième ($B[4] = B[5] = 0$). Un mauvais choix aboutit à un classement consensuel de très mauvaise qualité, parfois même à l'opposé de ce qui est théoriquement attendu.

4.6.3 Influence des vecteurs de pénalité sur la capacité à partitionner

Partitionner le problème initial en sous-problèmes indépendants est un défi important puisque c'est ce qui permet l'utilisation d'un algorithme exact lorsque le nombre d'éléments est trop élevé. Nous nous intéressons ici à la capacité de notre algorithme ParCons à partitionner le problème initial en de nombreux sous-problèmes selon les valeurs des vecteurs de pénalité. Les expériences décrites ici font l'objet d'un article en cours de rédaction.

Intuitivement, s'il y a une position i dans les vecteurs telle que $T[i]$ est trop petit par rapport à $B[i]$, on va augmenter le nombre de paires pour lesquelles la relation de préférence est l'égalité et ainsi augmenter la connexité du graphe. Il sera en conséquence plus difficile de partitionner le problème initial en de nombreux sous-problèmes.

Influence de B[6] sur le partitionnement.

$B[6]$ est un paramètre qui demande une attention particulière, nous en détaillons les raisons ci-dessous.

Interprétation de $B[6]$. Ce paramètre correspond au coût à payer si l'on veut placer un élément x avant un autre élément y dans le classement consensuel pour chaque classement d'entrée tel que x et y sont tous les deux absents. Ce paramètre est le seul qui différencie le score de Kemeny unifiant (Définition 4.4) et le score de Kemeny généralisé à la pseudo-distance (Définition 2.3.6). Les vecteurs de pénalité sont indiqués dans la Sous-section 4.2.2. Dans le premier cas, les éléments absents sont "virtuellement" placés *ex-aequo* à la fin du classement (pour simuler le comportement de la technique d'unification sans avoir à modifier les classements d'entrée). Une relation $x \prec_c y$ dans le classement consensuel c nécessite donc de payer un coût pour "rompre" l'égalité dans le classement d'entrée si x et y sont absents. Dans le deuxième cas, au contraire, on considère que les éléments absents n'ont pas de raison d'être *ex aequo*. En conséquence, les classements d'entrée tels que x et y sont tous les deux absents n'induisent pas de coût pour la paire (x, y).

Intuition quant à l'influence de $B[6]$ sur la capacité de ParCons à partitionner le problème initial en sous-problèmes. Considérons une paire $x \neq y \in U$. Dans le graphe G_e (Définition 3.4.1) à la base du partitionnement, il y a à la fois un arc de x vers y et un arc de y à x si et seulement si la relation la moins coûteuse est uniquement celle de mettre x et y à égalité dans le classement consensuel. Un tel cas de figure a tendance à diminuer le nombre de composantes fortement connexes du graphe (donc le nombre de sous-problèmes) puisqu'il augmente sa connexité. Lorsque des classements d'entrée sont très incomplets comme c'est le cas pour les jeux de données du *Dataset 1*, le fait de choisir une valeur de 1 plutôt que 0 pour $B[6]$ va augmenter le coût des relations "avant" pour de nombreuses paires, sans augmenter le coût des relations "égalité". On s'attend donc à ce que le nombre de sous-problèmes soit beaucoup plus petit avec $B[6] = 1$ qu'avec $B[6] = 0$.

Expérience (iv) : variations de $B[6]$ et nombre de sous-problèmes obtenus par ParCons. Pour cette expérience, on utilise le *Dataset 1* correspondant aux jeux de données biologiques (reformulations de maladies). Cette expérience a été menée en utilisant le code source du logiciel ConQuR-BioV2 contrairement aux autres expériences qui ont été menées en utilisant le code source du logiciel CoRankCo. On fait varier le paramètre $B[6]$ de 2 à 0 par pas de 0.1. Étant donnée une valeur de $B[6]$, pour chaque jeu de données, on divise le nombre moyen de sous-problèmes obtenus par le nombre d'éléments (nombre maximal possible de sous-problèmes que l'on peut obtenir). On associe alors à $B[6]$ la moyenne des valeurs ainsi calculées. Plus cette moyenne est proche de 1 (respectivement de 0), plus (respectivement moins) l'algorithme ParCons a pu partitionner le problème initial en de nombreux sous-problèmes. Les coefficients fixes et variables de (B, T) de cette expérience sont rappelés dans la Table 4.4.

Le résultat est présenté en Figure 4.6.

Résultat de l'expérience. On observe sur la Figure 4.6 que la capacité de ParCons à partitionner le problème initial en sous-problèmes décroît lorsque $B[6]$ augmente. En effet, lorsque $B[6] = 0$, le nombre de sous-problèmes

B	0	1	1	0	1	**B[6]**
T	1	1	0	1	1	0

TABLE 4.4 – Vecteurs de pénalité considérés pour l'expérience (iv)

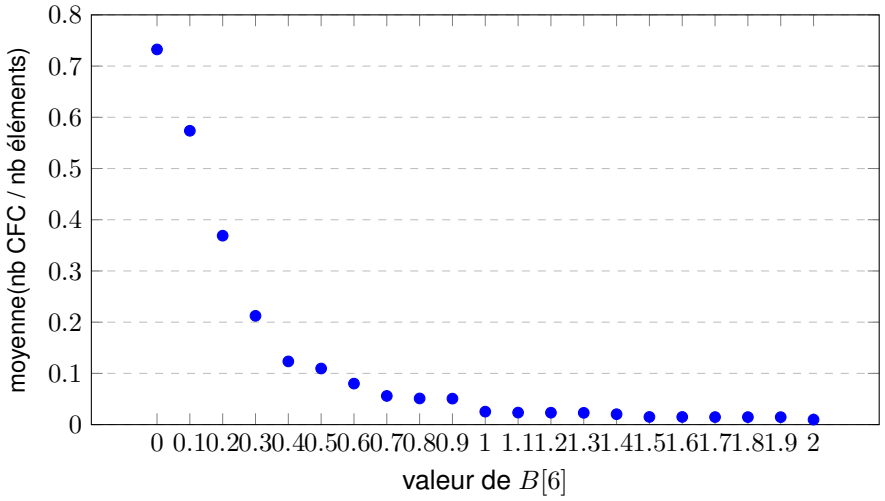


FIGURE 4.6 – Influence de $B[6]$ sur la capacité à partitionner le problème initial en sous-problèmes : moyenne sur l'ensemble des jeux de données du Dataset 1 du nombre de sous-problèmes obtenus par l'algorithme ParCons divisé par le nombre d'éléments en fonction de $B[6]$.

obtenus correspond en moyenne à 72% du nombre d'éléments. Ce chiffre tombe à 11% lorsque $B[6] = 0.5$ et à 3% lorsque $B[6] = 1$.

Conclusion de l'expérience.

L'impact de $B[6]$ est particulièrement flagrant sur la capacité de ParCons à diviser le problème initial en sous-problèmes. Une valeur de 0 permet un partitionnement en de nombreux sous-problèmes, mais cette capacité à partitionner décroît très vite lorsque $B[6]$ augmente. La qualité du classement consensuel vis-à-vis du score associé devient alors difficile à évaluer dans la mesure où la quasi-totalité des éléments sont placés dans le classement consensuel par une heuristique. Une étude plus approfondie sur les données valide l'intuition décrite ci-dessus : le nombre de paires d'éléments dont le coût le moins onéreux est celui de les mettre à égalité croît fortement avec la valeur de $B[6]$.

Influence du vecteur T sur la capacité à partitionner.

Nous avons rappelé lors de l'expérience précédente que si pour deux éléments $x \neq y$, le coût du positionnement relatif le plus faible correspond (uniquement) à la relation d'égalité, la connexité du graphe G_e augmente ce qui a pour conséquence de diminuer nos chances d'obtenir un grand nombre de composantes fortement connexes. Or, le vecteur T correspond aux coûts de mettre deux éléments x et y à égalité dans le classement consensuel selon les positionnements relatifs de x et y dans les classements d'entrée. On peut facilement imaginer que des coefficients trop faibles dans le vecteur T vont tendre à favoriser les égalités, réduisant ainsi la capacité de ParCons à diviser le

problème initial en sous-problèmes.

Expérience (v) : influence de $T[1]$ et $T[2]$ sur le nombre de sous-problèmes obtenus par ParCons. Comme dans l'expérience précédente, nous utilisons le *Dataset 1* correspondant aux données biologiques. On fait ici varier $T[1]$ et $T[2]$. Ces valeurs correspondent au coût à payer, si l'on veut mettre x et y à égalité dans le classement consensuel, pour chaque classement d'entrée tel que x est avant y ou y avant x. Pour rappel, par définition d'une KCF, on a $T[1] = T[2]$. Nous faisons varier $T[1]$ de 0 à 1 par pas de 0.1 (les valeurs de $T[4]$ et $T[5]$ sont fixées à 1). A nouveau, on regarde le nombre moyen de sous-problèmes obtenus normalisé par le nombre d'éléments. Les coefficients fixes et variables de (B, T) de cette expérience sont rappelés dans la Table 4.5.

B	0	1	1	0	1	0
T	**T[1]**	**T[1]**	0	1	1	0

TABLE 4.5 – Vecteurs de pénalité considérés pour l'expérience (v)

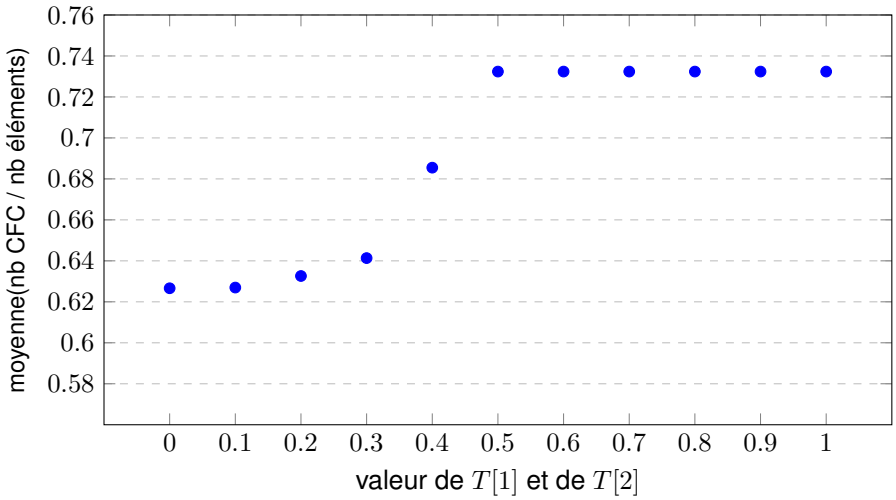


FIGURE 4.7 – Influence de $T[1]$ et $T[2]$ sur la capacité à partitionner le problème initial en sous-problèmes : moyenne sur l'ensemble des jeux de données du Dataset 1 du nombre de sous-problèmes obtenus par l'algorithme ParCons divisé par le nombre d'éléments en fonction de $T[1] = T[2]$.

Résultat de l'expérience. Les résultats sont présentés en Figure 4.7. On voit que le nombre de sous-problèmes augmente avec $T[1]$ ce qui est en adéquation avec le résultat attendu. Cependant, on observe que même avec $T[1] = 0$, on garde un nombre de sous-problèmes plutôt important. Ce résultat est, cette fois, moins attendu.

Hypothèse pour expliquer le résultat de l'expérience. Le fait que le nombre de sous-problèmes reste relativement élevé même avec des valeurs nulles pour $T[1]$ et $T[2]$ peut s'expliquer. En effet, les données biologiques sont particulièrement incomplètes. Les paramètres $T[4]$ et $T[5]$ (coût à payer si l'on veut mettre x et y à égalité dans le classement consensuel pour chaque classement d'entrée tel que x ou y est présent mais pas les deux) étant fixés à 1, ils peuvent compenser les faibles valeurs de $T[1]$ et $T[2]$ et ainsi maintenir des coûts élevés pour les égalités, ce qui est propice à un plus grand nombre de composantes fortement connexes pour G_e.

4.6.4 Conclusions des expériences menées

Plusieurs points ressortent de ce chapitre et sont listées ci-dessous.

- Le choix des paramètres du modèle est particulièrement important si les classements à agréger correspondent à des données réelles. Dans un tel cas, il convient de comprendre les raisons pour lesquelles les classements sont incomplets afin de choisir des paramètres pertinents. En particulier, la valeur de $B[5] - B[4]$ doit être nulle si les éléments manquants sont incomparables avec les éléments présents. Une valeur de 1 est conseillée pour les cas où les éléments manquants doivent être considérés comme moins pertinents que les éléments présents.
- Il est préférable de faire des choix francs. Des pénalités strictement supérieures à 0 mais proches de 0 aboutissent à des résultats de qualité moyenne.
- Choisir une KCF pertinente vis-à-vis du contexte est d'autant plus important que les jeux de données sont dissimilaires.
- Certains paramètres comme $B[6]$, $T[1]$ (égal à $T[2]$) et $T[4]$ (égal à $T[5]$) ont une influence sur la capacité de ParCons à partitionner le problème initial en sous-problèmes. Des valeurs trop basses pour les valeurs de T par rapport aux valeurs de B (pour un même indice) ont tendance à favoriser les égalités, augmentant ainsi la connexité du *graphe des éléments* (Définition 3.4.1).

4.7 Conclusion

Dans ce chapitre, nous avons introduit un modèle paramétré par deux vecteurs de pénalité de taille 6 qui inclut les principales adaptations du score de Kemeny aux classements incomplets que l'on peut trouver dans la littérature (en particulier les adaptations définies dans [4, 17, 19, 24, 44, 78]) ainsi que le score de Kemeny généralisé de Fagin *et al.* pour les égalités [47] mais également d'en concevoir de nouvelles. Nous avons appelé KCF (pour *fonction Kemeny compatible*) une fonction rentrant dans le cadre de notre modèle, et démontré que les adaptations du score de Kemeny citées précédemment sont des KCFs. Le choix des valeurs des paramètres est lié au sens que l'on veut donner aux données manquantes. Le contexte lié aux données et en particulier la raison pour laquelle certains éléments sont manquants dans certains classements est un élément clé pour guider l'utilisateur vers le choix des paramètres. Par exemple, on ne choisira pas les mêmes vecteurs de pénalité selon que les classements correspondent à des films notés par des utilisateurs sur une plateforme ou à des gènes associés à une maladie. Dans le premier cas, parmi l'ensemble des films disponibles, un utilisateur ne note que les films qu'il a déjà visionnés, rendant impossible la comparaison entre les films qu'il a noté et les films non notés. Dans le deuxième cas, un gène absent d'un classement signifie que ce gène est considéré comme non pertinent vis-à-vis du nom de maladie utilisé pour la requête.

Une étude axiomatique de ce modèle, inspirée par les critères d'équité de la théorie du choix social, met en avant que certaines valeurs de paramètres peuvent offrir des garanties. En particulier, certains de ces critères comme le critère de Condorcet et le LIIA couramment utilisés dans le cadre des classements complets et sans égalités peuvent être redéfinis en utilisant les paramètres du modèle, aboutissant à des techniques de partitionnement du problème initial en sous-problèmes indépendants sur les classements incomplets et avec égalités.

Les algorithmes classiquement utilisés dans le cadre de l'agrégation de classements et compatibles avec le modèle paramétré ont été adaptés, tout comme l'ensemble de nos contributions du Chapitre 3 (en particulier les algorithmes ParCons et ParFront qui ont été généralisés à n'importe quelle KCF). Un outil en ligne, CoRankCo, a été créé afin que les utilisateurs puissent calculer un classement consensuel pour des classements possiblement incomplets et avec égalités en choisissant leur algorithme ainsi que le vecteur de pénalité le plus adapté au contexte. Un *package* python du même nom a également été développé et mis à disposition de la communauté scientifique.

Une évaluation du modèle a permis de constater que la qualité des classements consensuels renvoyés par ConQuR-BioV2 est due (entre autres) à l'adéquation entre le contexte biologique et le choix du score de Kemeny adapté à la pseudo-distance. En effet, alors que ConQuR-BioV2 renvoie des classements de gènes qui correspondent mieux à une réalité biologique que la base de données *Gene* du NCBI (voir Section 3.9), nous avons montré à travers une expérience qu'un mauvais choix de vecteurs aboutit au calcul d'un classement consensuel de plus mauvaise qualité que le classement obtenu par la base de données *Gene* du NCBI qui ne se sert pas des reformulations. Plus généralement, les expériences menées avec les logiciels ConQuR-BioV2 et CoRankCo montrent que l'utilisation d'une adaptation du score de Kemeny (donc une KCF) pour les données manquantes peut aboutir à des résultats d'une piètre qualité si le contexte expliquant les données manquantes n'est pas pris en compte. Ainsi, les valeurs des vecteurs de pénalité du modèle défini au Chapitre 4 ont une influence très importante sur la qualité du résultat. Nous avons enfin pu observer que la capacité de ParCons à partitionner le problème initial en sous-problèmes est également dépendante de certaines valeurs de ces coefficients. Les valeurs de $T[2]$, $T[4]$, $T[5]$ et $T[6]$ ne doivent pas être trop faibles par rapport à $B[2]$, $B[4]$, $B[5]$ et $B[6]$ au risque de ne plus pouvoir partitionner le problème initial en sous-problèmes.

Résumé du Chapitre 4

De nombreux jeux de données réels correspondent à des classements incomplets et avec égalités. Plusieurs généralisations du score de Kemeny ont été proposées [4, 17, 19, 24, 44, 78]. Le problème est de savoir, étant donné une liste de classements, quelle généralisation choisir parmi celles citées. En effet, aucune information n'est donnée quant aux caractéristiques des données qui rendent pertinentes l'utilisation de ces différentes généralisations. Par ailleurs, ces généralisations ne sont pas associées à un algorithme exact ou a des propriétés susceptibles d'amener une heuristique à calculer un classement consensuel de bonne qualité.

Pour répondre à ces enjeux, nous avons défini un modèle paramétré permettant d'apporter un regard qualitatif sur les données. Plus précisément, les paramètres du modèle sont deux vecteurs réels B et T de taille 6 dont le premier (respectivement le deuxième) correspond aux coûts pour chaque paire d'éléments (x, y) telle que x est avant y (respectivement x est à égalité avec y) dans le classement consensuel des 6 situations suivantes pour un classement d'entrée r : x est avant y, x est après y, x est à égalité avec y, x est présent et y absent, x est absent et y présent, x et y sont absents. Nous avons appelé KCF (*Kemeny Compatible Function*) une "fonction d'agrégation" qui rentre dans le cadre de notre modèle et avons démontré que les généralisations du score de Kemeny citées au paragraphe précédemment sont des KCFs (les vecteurs de pénalités associés à chacune ont été déterminés).

Sur les aspects algorithmiques, dans le but de fournir à l'utilisateur un classement consensuel de bonne qualité en un temps raisonnable, nous avons (i) conçu et implémenté un algorithme exact en PLNE, (ii) généralisé l'ensemble des contributions du chapitre précédent à notre modèle afin d'utiliser ParCons et ParFront sur n'importe quelle KCF, (iii) généralisé plusieurs heuristiques couramment utilisées dans le cadre de l'agrégation de classements pour qu'elles considèrent les vecteurs de pénalité du modèle. Les heuristiques basées sur des comparaisons par paires comme BioConsert [32], KwikSort [3] et Copeland [36] s'adaptent naturellement au modèle, tandis que d'autres heuristiques comme Borda [39], MedRank [46] ne peuvent s'adapter qu'à certaines KCFs bien déterminées.

Nous avons également fait une étude axiomatique de notre modèle en déterminant des conditions nécessaires et suffisantes sur les vecteurs de pénalités pour que le critère de majorité, le critère de Condorcet et le critère d'indépendance locale des alternatives non pertinentes particulièrement importants dans la théorie du choix social soient respectés. L'intérêt de respecter ou non ces critères a été mis en lien avec les données réelles.

Des expériences ont été menées sur des données biologiques et synthétiques pour évaluer le modèle et ont abouti à deux enseignements principaux. Premièrement, il est pertinent de choisir une valeur de 1 pour $B[5] - B[4]$ si les éléments absents doivent être pénalisés vis-à-vis des présents, et 0 sinon. Plus les classements sont dissimilaires et incomplets, plus la valeur $B[5] - B[4]$ influe sur le classement consensuel calculé. Deuxièmement, les paramètres $B[6]$, $T[1]$ et $T[2]$ ont une influence sur la capacité de ParCons à partitionner le problème initial en sous-problèmes. L'outil en ligne CoRankCo a été implémenté (`https://corankco.lri.fr/`). Il permet à un utilisateur d'agréger des classements incomplets et avec égalités. Un *package* python du même nom est également disponible.

Conclusion

Nous sommes partis d'un besoin réel exprimé par nos collaborateurs biologistes et médecins, besoin rencontré dans leur tâche de recherche d'information : celui de rendre compte des points communs et des différences entre plusieurs classements de gènes obtenus en interrogeant une même base de données. Plus précisément, lorsqu'un médecin ou un biologiste cherche à connaître les gènes les plus associés à une maladie donnée, il doit utiliser plusieurs synonymes de la maladie d'intérêt pour faire apparaître bien positionnés certains gènes connus pour être étroitement liés à la maladie [24]. Il est alors difficile de conclure quant aux gènes les plus impliqués face à l'information apportée par plusieurs classements (possiblement quelques dizaines dans ce contexte) de plusieurs milliers de gènes. Une difficulté supplémentaire est que les classements peuvent contenir des égalités (certains gènes peuvent être *ex aequo* dans un classement, par exemple s'ils sont d'égale importance vis-à-vis de la maladie) et être incomplets (certains gènes ne sont présents que dans certains classements).

Nous nous sommes alors intéressés à la problématique d'**agrégation de classements** en considérant la méthode Kemeny-Young [59, 86] qui consiste à déterminer un classement consensuel de score de Kemeny minimal. Ce problème NP-difficile dans la majorité des cas [12, 20, 44] intéresse de nombreuses communautés, en particulier la communauté algorithmique [3, 17, 44, 69] et la communauté théorie du choix social [9, 10, 26]. Ces deux communautés sont à l'origine de la grande majorité des résultats théoriques majeurs concernant ce problème [17, 83, 86]. Un objectif récurrent est de concevoir des techniques de réduction d'espace [69], consistant le plus souvent à partitionner le problème de départ en sous-problèmes indépendants [17, 83]. Ces méthodes sont souvent utilisées pour diminuer le temps d'exécution d'un algorithme exact, mais sont rarement suffisantes sur des instances de grande taille. En conséquence, de nombreux algorithmes d'approximation et heuristiques sont utilisés [2, 3, 32, 36, 39, 46].

Capacité à gérer les classements avec égalités et les classements incomplets. La majorité de ces heuristiques et algorithmes d'approximation ont été conçus pour les classements complets et sans égalités [2, 3, 36, 39, 46]. Ils ont été adaptés dans [25, 23] pour gérer les classements avec égalités. Bien que plusieurs adaptations du score de Kemeny aient été proposées pour gérer les classements incomplets [4, 17, 24, 44], aucune n'est associée à des techniques de partitionnement ou à d'autres résultats théoriques permettant de calculer un classement consensuel optimal sur des instances de grande taille. Seules des heuristiques sont utilisées [4, 24], et n'apportent

par essence aucune garantie sur la qualité des classements consensuels calculés.

Première contribution : une heuristique basée sur une technique de partitionnement en sous-problèmes indépendants. Notre première contribution publiée dans [7, 8] est d'avoir établi une condition suffisante permettant de partitionner le problème initial en sous-problèmes indépendants de sorte que la concaténation d'un classement consensuel optimal de chaque sous-problème forme un classement consensuel optimal pour le problème de départ. Cette propriété repose sur la représentation en graphe des classements de départ. Nous avons implémenté l'algorithme ParCons dont la première étape consiste à effectuer le partitionnement en sous-problèmes. La deuxième étape consiste à appeler un algorithme exact sur les sous-problèmes non triviaux dont la taille permet de le faire (en pratique, moins de 80 éléments), ou à appeler une heuristique pour les sous-problèmes de grande taille. Plusieurs expériences ont montré l'efficacité de ParCons sur un ensemble de 1354 jeux de données biologiques de grande taille (entre 100 et 1557 éléments) : un classement consensuel optimal a été trouvé sur 60 % des jeux de données, dont la moitié n'a eu besoin que du partitionnement (tous les sous-problèmes obtenus étaient triviaux). Nous avons également montré la pertinence de notre approche en utilisant des *gold standards* : la base de données Orphanet spécialisée dans les maladies rares fournit pour une maladie donnée un ensemble (généralement petit) de gènes connus pour être particulièrement associés à une maladie donnée. Nous avons observé à travers plusieurs expériences que le classement consensuel renvoyé par ParCons met mieux en avant ces gènes que le classement de la base de données *Gene* du NCBI n'utilisant pas les reformulations synonymes. L'algorithme ParCons est implémenté dans l'outil en ligne ConQuR-BioV2 destiné aux médecins et biologistes et dont le fonctionnement est décrit dans la Section 3.7.

Perspectives de cette première contribution. Nous dégageons deux perspectives principales à cette première contribution. La *propriété de concaténation* (Propriété 3.5.1) à la base de l'heuristique ParCons garantit que concaténer un classement consensuel optimal de chaque sous-problème forme un classement consensuel optimal pour le problème de départ. Ces sous-problèmes sont obtenus en calculant les composantes fortement connexes du *graphe des éléments* (Définition 3.4.1) qui représente les classements d'entrée.

Partitionner les composantes fortement connexes. Une première perspective est d'établir de nouvelles conditions suffisantes, plus souples, qui permettraient de partitionner le sous-problème formé par une unique composante fortement connexe en plusieurs sous-problèmes. Trouver de telles propriétés serait particulièrement utile dans la mesure où plusieurs jeux de données biologiques parmi les 1354 qui ont servi à notre étude expérimentale ont abouti au calcul d'une seule composante fortement connexe.

Définir d'autres graphes. Par ailleurs, dans le *graphe des éléments*, un arc de x vers y veut dire que le positionnement relatif le moins coûteux pour la paire (x, y) dans le classement consensuel n'est pas de mettre y avant x. Un tel arc peut correspondre à trois situations différentes : le coût du positionnement relatif le plus bas

peut être celui de mettre x avant y, x à égalité avec y, ou les deux. Représenter des classements incomplets avec égalités par un graphe dont la définition des arcs est différente pourrait apporter des informations complémentaires qu'il serait intéressant d'exploiter.

Deuxième contribution : donner à l'utilisateur un indicateur de robustesse du classement consensuel. Notre deuxième contribution, publiée dans [8], est d'avoir établi une condition suffisante capable de mettre en évidence des points communs entre tous les classements consensuels optimaux. Plus précisément, nous calculons à l'aide du *graphe des éléments* une partition ordonnée des éléments à trier telle que tous les classements consensuels optimaux respectent (au sens de la Définition 3.5.1) cette partition. Cette dernière détermine des *frontières* : une frontière en position k indique que l'ensemble des éléments constituant le top-k est le même pour tous les classements consensuels optimaux. Nous avons implémenté l'algorithme ParFront dans ConQuR-BioV2 afin de rendre ces frontières visibles pour la communauté biologique et médicale. En effet, l'information apportée par les frontières peut être particulièrement utile à l'utilisateur qui aura une confiance plus élevée dans les choix de gènes à explorer si des frontières apparaissent aux premières positions du classement consensuel renvoyé. Notre étude expérimentale a montré que les classements issus de reformulations synonymes des maladies permettent d'obtenir un nombre de frontières non négligeable, en particulier aux premières positions. Nous avons également démontré que notre approche permet de calculer toutes les frontières obtenues par les éléments propres [17] et le critère de Condorcet étendu [83], et observé expérimentalement que nous obtenons beaucoup plus de frontières qu'avec ces deux dernières méthodes.

Perspectives de cette deuxième contribution. Un inconvénient de notre approche est qu'elle repose sur une condition suffisante. Ainsi, sur certaines instances pour lesquelles il y a un seul classement consensuel optimal, elle ne permet de calculer aucune frontière. Or, sur ces instances, il est tout aussi important d'indiquer à l'utilisateur si le classement consensuel renvoyé est fiable ou non. De plus, les frontières ne permettent pas de savoir s'il suffirait de modifier légèrement les classements d'entrée pour aboutir au calcul d'un classement consensuel très différent. Il serait particulièrement utile pour le biologiste de savoir à quel point les premiers éléments du classement consensuel ont une position robuste vis-à-vis de petites modifications dans les classements d'entrée. Par ailleurs, le logiciel ConQuR-BioV2 repose sur la recherche de synonymes d'une maladie donnée. Or, les médecins sont susceptibles de vouloir refuser certains synonymes s'ils leurs paraissent peu pertinents. À nouveau, la question de savoir à quel point le classement consensuel peut être robuste vis-à-vis de la perte d'un ou plusieurs classements se pose.

Utiliser des outils statistiques. Une piste à étudier pour tenter de répondre à ces problèmes est d'utiliser des outils statistiques afin de calculer des intervalles de confiance sur la position des éléments. Des méthodes bootstrap ont déjà été utilisées sur des classements complets et sans égalités [51, 87]. Il serait intéressant d'étudier

à quel point le modèle utilisé dans ce cadre peut se généraliser aux classements incomplets et avec égalités. Nous avons débuté son investigation lors d'un stage de M2 co-encadré avec Marie-Anne Poursat du LMO (Laboratoire de Mathématiques d'Orsay) et les résultats préliminaires obtenus sur des permutations sont encourageants. Par ailleurs, de nombreux travaux en apprentissage statistique concernent l'agrégation de classements [1, 27, 42, 63] et peuvent être utilisés dans le but de quantifier l'incertitude. L'utilisation de forêts aléatoires peut aider à calculer un score d'importance associé à un élément [50, 54], reflétant le niveau de confiance qu'on peut avoir en sa position dans le classement consensuel.

Exploiter l'information apportée par les tris topologiques. Enfin, les tris topologiques du graphe des composantes fortement connexes que l'on calcule peuvent également donner des informations quant à la robustesse du classement consensuel renvoyé. En effet, nous avons montré que chaque tri topologique est associé à un classement consensuel optimal distinct. Considérons une composante fortement connexe du *graphe des éléments* qui serait en troisième position d'un premier tri topologique et en cinquantième position d'un deuxième tri topologique, on saurait alors que les éléments qui la constituent ont une position qui n'est pas du tout robuste dans le classement consensuel renvoyé.

Troisième contribution : un modèle pour gérer les classements incomplets. Notre troisième contribution concerne la gestion des classements incomplets et fait l'objet d'un article en cours de rédaction. Nous avons conçu un modèle paramétré incluant les différentes généralisations du score de Kemeny de la littérature pour gérer les classements incomplets [4, 19, 17, 24, 44, 78] et les classements avec égalités [47], et permettant d'en concevoir de nouvelles. Une fonction rentrant dans le cadre de notre modèle est appelée KCF (pour *fonction Kemeny compatible*). L'intérêt de ce modèle est d'apporter un regard qualitatif sur les données et particulièrement sur l'interprétation des données manquantes, palliant le manque d'information concernant les cas d'utilisation des généralisations du score de Kemeny de la littérature. Il est en effet possible, entre autres, de décider à quel point les éléments manquants doivent être pénalisés vis-à-vis des éléments présents étant donné le contexte associé aux données. Les adaptations [4, 19, 17, 24, 44, 78] n'étant associées à aucune technique de réduction d'espace ni aucun algorithme exact, nous avons alors conçu et implémenté un algorithme exact en programmation linéaire en nombre entiers capable de renvoyer l'ensemble des classements consensuels optimaux quelle que soit la KCF considérée (quelles que soient les valeurs choisies pour les paramètres), et généralisé plusieurs heuristiques pour qu'elles prennent en compte les paramètres du modèle. En particulier, les méthodes présentées au Chapitre 3 (les algorithmes ParCons et ParFront) ont été généralisées pour être utilisées dans le cadre de ce modèle. Une étude axiomatique a été effectuée en déterminant des conditions nécessaires et suffisantes sur les valeurs des vecteurs pour que certains critères d'équité de la théorie du choix social soient respectés. Ces critères ont été mis en lien avec les données réelles. Une étude expérimentale a été menée afin d'évaluer le modèle proposé. Nous avons montré, entre autres, qu'avec une généralisation du score de Kemeny non pertinente vis-à-vis du contexte des reformulations synonymes

de maladies, le classement consensuel renvoyé par ConQuR-BioV2 devient très souvent moins bon que celui renvoyé par la base de données *Gene* du NCBI pour le même mot-clé choisi. Un lien a également été établi entre certains paramètres du modèle et la capacité de ParCons à partitionner le problème principal en sous-problèmes. Pour terminer, nous avons implémenté l'outil en ligne CoRankCo (`https://corankco.lri.fr/`) permettant à un utilisateur d'agréger des classements incomplets et avec égalités. Un *package* python du même nom est également disponible.

Perspectives de cette troisième contribution. Les perspectives sont nombreuses pour cette contribution qui fait l'objet d'un article en cours de rédaction. Tout d'abord, l'intégralité des perspectives associées aux deux premières contributions sont également des perspectives pour cette troisième contribution. En effet, il serait particulièrement utiles que les futurs travaux associés aux perspectives décrites précédemment puissent prendre en compte la généricité du modèle présenté ici. D'autres perspectives sont envisagées.

Approfondir l'évaluation du modèle. Deux paramètres n'ont pas fait l'objet d'expériences. Le premier (respectivement le deuxième) correspond au coût à payer pour placer un élément x à égalité avec un autre élément y dans le classement consensuel lorsque dans un classement d'entrée, un des deux éléments est absent (respectivement x et y sont tous les deux absents). Des expériences pourraient par exemple mettre en avant le lien entre les données manquantes, les valeurs de ces paramètres et la capacité de ParCons à partitionner le problème initial en sous-problèmes.

Compléter l'étude axiomatique. Les critères d'équité de la théorie du choix social peuvent avoir une importance majeure pour le calcul de classements consensuels. Nous avons établi des conditions nécessaires et suffisantes pour plusieurs d'entre eux mais d'autres critères d'équité mériteraient d'être étudiés dans le cadre de notre modèle. On peut en particulier citer le *Pareto criterion* (si dans tous les classements d'entrée on a x avant y, alors x doit être avant y dans le classement consensuel), *Monotonicity* (si un élément x voit sa position améliorée dans certains classements sans que les autres éléments ne soient déplacés, alors la position de x dans le classement consensuel ne peut pas empirer) et le *Consistency* (si on divise les classements d'entrée en deux groupes et que l'ensemble des éléments en première position est identique pour les deux sous-groupes, alors cet ensemble doit être conservé si l'on réunit les sous-groupes).

Intégrer les techniques de réduction d'espace de Robin Milosz [68]. Robin Milosz et Sylvie Hamel ont établi des techniques de réduction d'espace qui permettent de garantir, lorsque certaines conditions nécessaires sont respectées, qu'un élément x est avant un autre élément y dans tous les classements consensuels optimaux. Ces travaux publiés dans [69] ont été menés sur des classements complets et sans égalités. Intégrer ces techniques de réduction d'espace pourrait diminuer le temps d'exécution de notre algorithme exact sur des instances de grande taille.

Perspective générale : travailler sur des données réelles de natures différentes. Les classements de gènes issus des reformulations de maladies ont certaines particularités qui peuvent jouer un rôle important pour l'obtention de frontières. Par exemple, ces jeux de données contiennent généralement peu d'éléments à égalité. Regarder le nombre de frontières que l'on obtient sur des jeux de données de nature différente permettrait de mieux caractériser les jeux de données sur lesquels nos approches sont efficaces. Nous explicitons ci-dessous deux cas d'utilisation qu'il serait pertinent de considérer.

Données multi-omiques. Nous souhaitons investiguer davantage les techniques d'agrégation de classements dans un autre contexte biologique : les classements issus d'expériences multi-omiques qui considèrent des entités de natures différentes (gène, ARN ou protéine). Nos collègues biologistes et bioinformaticiens Gaëlle Lelandais et Thomas Denecker (Institut de biologie intégrative de la cellule, I2BC) utilisent ce type de données sur la problématique de l'identification des gènes sous-tendant l'homéostasie du fer chez la levure *Candida glabrata* [40, 41]. Les différents classements obtenus correspondent à des conditions expérimentales différentes (variation de la température et de la concentration en fer dans le milieu, variation du pH). Les éléments sont classés selon un critère statistique (p-valeur, logFC, nombre de lectures cartographiées). Par exemple, plus l'expression d'un gène a été modifiée dans une condition expérimentale donnée par rapport à une condition témoin, plus la valeur absolue du logFC associé au gène est grande. L'utilisation des techniques d'agrégation de classements pourrait permettre de combiner l'information apportée par les différents classements donc les différentes conditions expérimentales et ainsi mettre en évidence des entités particulièrement impliquées dans le processus biologique étudié. Il faut veiller à l'impact des biais expérimentaux dans la comparaison des classements obtenus. En effet, un protocole expérimental de dosage protéique basé sur l'affinité avec l'eau ne permettra pas de repérer la présence de protéines hydrophobes. Ces protéines seront ainsi absentes des classements quelque soit leur concentration et leur rôle dans le métabolisme. Pourtant, l'expression du gène codant la protéine pourra être mis en évidence par séquençage des ARN. Dans ce contexte, il est important de ne pas pénaliser les éléments manquants vis-à-vis des présents puisque les données manquantes sont dues à un biais de protocole.

Des premiers résultats encourageants ont été obtenus dans le cadre du stage de Master 2 de Joffrey Benoist qui a appliqué des techniques d'agrégation de classements sur les classements issus des expériences multi-omiques. Lors de leurs expériences, nos collègues avaient identifié 214 gènes clés dans l'homéostasie du fer pour un ensemble de 637 gènes considérés. L'application d'une technique d'agrégation de classements a abouti au calcul d'un classement consensuel prometteur : 72 de ces gènes clés ont figuré dans le top-100.

Données issues de notes d'utilisateurs sur des plateformes. Si l'on souhaite sortir du cadre de la bioinformatique, on pourrait étudier le cas des plateformes sur lesquelles les utilisateurs sont invités à donner des notes (généralement entre 0 et 5, par pas de 0.5 ou de 1). C'est par exemple le cas des plateformes contenant des bases de données de films/séries. Les données fournies par Netflix dans le cadre du *prix Netflix* [15] (accessibles publiquement) pourraient être intéressantes à étudier dans le cadre de l'agrégation de classements. Dans un tel

contexte, chaque classement est associé à un utilisateur et les éléments dans un classement donné sont les films notés par l'utilisateur. Ces classements ont deux particularités : (i) les égalités sont très nombreuses puisque le nombre de notes possibles est restreint et (ii) les classements sont particulièrement incomplets puisqu'un même utilisateur a vu un très petit nombre de films par rapport au nombre de films disponibles. Il serait intéressant de regarder si certains paramètres du modèle présenté au Chapitre 4 permettent d'aboutir à des classements consensuels de bonne qualité sur de tels jeux de données, et si ces caractéristiques particulières ont une influence sur les algorithmes ParCons et ParFront.

Annexe A

Calcul efficace de $S^{(B,T)}$ (Définition 4.2.1)

Cette annexe a pour but de donner une intuition sur le fonctionnement d'un algorithme permettant de calculer efficacement le score de Kemeny généralisé au modèle paramétré présenté dans le chapitre 4 (voir Définition 4.2.1). Cet algorithme étend celui qui calcule le score de Kemeny avec une complexité $O(|R| * |U| * log(|U|)$ inspiré du tri fusion (utilisation de l'algorithme permettant de compter le nombre d'inversions dans un tableau d'entiers) [44].

Pour rappel, le *score de Kemeny* $S^{(B,T)}$ *avec vecteurs de pénalité* B *et* T est défini de la manière suivante (Définition 4.2.1) : pour tout $(c, R) \in C(U) \times A(U)^{<\infty}$,

$$S^{(B,T)}(c,R) = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \langle B, \Omega^R_{x,y} \rangle \; + \; \frac{1}{2} * \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} \langle T, \Omega^R_{x,y} \rangle \tag{A.1}$$

avec $\Omega^R_{x,y}$ le vecteur $(\omega^R_{x \prec y}, \omega^R_{y \prec x}, \omega^R_{x \equiv y}, \omega^R_{x.\overline{y}}, \omega^R_{y.\overline{x}}, \omega^R_{\emptyset})$, avec

- $\omega^R_{x \prec y} = |\{r \in R : x \prec_r y\}|$ soit le nombre de classements de R tels que x est avant y,
- $\omega^R_{x \equiv y} = |\{r \in R : x \equiv_r y\}|$ soit le nombre de classements de R tels que x et y sont à égalité,
- $\omega^R_{x.\overline{y}} = |\{r \in R : x \diamond_r y\}|$ soit le nombre de classements de R tels que x est présent alors que y ne l'est pas,
- $\omega^R_{\emptyset} = |\{r \in R : x \notin dom(r) \wedge y \notin dom(r)\}|$ soit le nombre de classements de R tels que ni x ni y ne sont présents.

où $\langle B, \Omega^R_{x,y} \rangle$ (respectivement $\langle T, \Omega^R_{x,y} \rangle$) correspond au produit scalaire de B (respectivement T) avec $\Omega^R_{x,y}$.

Les paragraphes ci-dessous ont pour objectif de donner une intuition sur la manière de calculer efficacement la valeur de $S^{(B,T)}$.

Remarque. Pour que les calculs présentés ci-dessous soient visuellement homogènes avec le code python donné à la fin de cette annexe, nous faisons commencer ici les listes et vecteurs à l'indice 0.

Étant donné un classement consensuel $c = [c_0, c_1, ..., c_{b-1}]$ contenant b *buckets* et une liste de classements R, pour calculer $S^{(B,T)}(c, R)$, on va calculer les deux vecteurs réels S_1 et S_2 de taille 6 définis de la manière suivante : $S_1 = \sum_{\substack{x \neq y \in U \\ x \prec_c y}} \Omega^R_{x,y}$ et $S_2 = \sum_{\substack{x \neq y \in U \\ x \equiv_c y}} \Omega^R_{x,y}$. On a alors $S^{(B,T)}(c, R) = \langle B, S_1 \rangle + \frac{1}{2} * \langle T, S_2 \rangle$.

L'astuce permettant de calculer S_1 et S_2 est la suivante : chaque classement r de R va être transformé de sorte que chaque élément de chaque *bucket* de r soit remplacé par la position de son *bucket* dans le classement c (pour rappel, c est un classement consensuel donc complet). De plus, pour tout entier $0 \leq i \leq b-1$, on va retenir les informations suivantes :

- le nombre d'éléments de $U \setminus dom(r)$ placés avant les éléments de c_i.
- le nombre d'éléments de $U \setminus dom(r)$ placés après les éléments de c_i.
- le nombre d'éléments de $U \setminus dom(r)$ qui font partie de c_i.

Illustration. Soit R une liste de classements, $c = [\{A\}, \{B, C, F, G\}, \{D\}, \{E\}, \{H\}]$ un classement consensuel et $r = [\{D\}, \{A, G, H\}, \{C, F\}]$ un classement de R. On observe que deux éléments de U n'apparaissent pas dans r : B et E. Pour calculer la valeur de $\sum_{\substack{x \neq y \in U \\ x \prec_c y}} \Omega^R_{x,y}$, on va considérer la liste de listes suivante : $r' = [[2], [0, 1, 4], [1, 1]]$. En effet, D est dans le bucket de c d'indice 2, A dans le bucket de c d'indice 0, G dans le bucket de c d'indice 1 et ainsi de suite. Pour la suite des calcul, il est important que toutes les listes de r' soient triées. La complexité de cette opération est $O(|U| * \log |U|)$ en utilisant un tri fusion.

Le tableau $T_1[0 \ldots b-1]$ ci-dessous donne, pour tout entier $0 \leq i \leq b-1$, le nombre d'éléments de $U \setminus dom(r)$ placés avant les éléments de c_i : $[0, 0, 1, 1, 2]$.

Le tableau $T_2[0 \ldots b-1]$ ci-dessous donne, pour tout entier $0 \leq i \leq b-1$, le nombre d'éléments de $U \setminus dom(r)$ placés après les éléments de c_i : $[2, 1, 1, 0, 0]$.

Le tableau $T_3[0 \ldots b-1]$ ci-dessous donne, pour tout entier $0 \leq i \leq b-1$, le nombre d'éléments de $U \setminus dom(r)$ qui font partie de c_i : $[0, 1, 0, 1, 0]$.

Notation. Pour une liste L d'entiers et un entier k, on note respectivement $card(L < k)$, $card(L > k)$ et $card(L = k)$ le nombre d'éléments de L qui sont respectivement supérieurs, inférieur et égaux à k.

Intuition pour le calcul de $S^{(B,T)}(c, R)$.

Afin d'illustrer la façon dont les vecteurs S_1 et S_2 peuvent être obtenus, nous allons montrer comment calculer la part d'un classement $r \in R$ dans le calcul final des vecteurs S_1 et S_2. Nous ne nous intéresserons pas au calcul de $S_1[0]$ ni de $S_2[2]$ dans la mesure où ces valeurs n'interviennent finalement pas pas dans le calcul du score de Kemeny (pour rappel, pour que $S^{(B,T)}$ soit une fonction Kemeny compatible, on doit avoir $B[0] = 0$ et $T[2] = 0$).

(i) Calcul de la valeur de $S_1[1]$ et de $S_2[0] + S_2[1]$ induite par $r \in R$. On sait que $S_1[1]$ est la somme pour chaque classement r de R du nombre de paires d'éléments (x, y) tels que x est avant y dans le consensus et y est avant x

dans r. De plus, $S_2[0]$ + $S_2[1]$ est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x est à égalité avec y dans le consensus et x est avant y ou y est avant x dans r.

Le calcul de $S_1[1]$ et de $S_2[0] + S_2[1]$ s'effectue pendant la procédure de fusion de listes. Soient L_1 et L_2 deux listes triées contenant respectivement n_1 et n_2 éléments. On suppose que L_1 provient de la fusion des listes $r'[u]$ à $r'[v]$, $v \geq u$, et que L_2 provient de la fusion des listes $r'[v+1]$ à $r'[w]$, $w \geq v+1$. Par construction, les éléments de L_1 sont avant les éléments de L_2 dans r. Ainsi, si un élément de L_2 est plus petit qu'un élément de L_1, on en déduit qu'il y a une inversion. La valeur de $S_1[1]$ induite par $r \in R$ vaut : $\sum_{0 \leq i \leq n_1 - 1} card(L_2 < L_1[i])$. De la même manière, la valeur de $S_2[0] + S_2[1]$ induite par $r \in R$ vaut $\sum_{0 \leq i \leq n_1 - 1} card(L_2 = L_1[i])$. Les listes étant triées, l'algorithme de fusion permet de calculer ces valeurs avec une complexité linéaire.

(ii) Calcul de la valeur de $S_1[2]$ induite par $r \in R$. On sait que $S_1[2]$ est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x est avant y dans le consensus et x et y sont à égalité dans r. On sait que deux éléments à égalité dans r sont nécessairement dans la même liste de r'. Il faut donc sommer pour chaque liste L de r' le nombre de paires d'éléments distincts de L (en tenant compte des redondances). Au final, la valeur de $S_1[2]$ induite par $r \in R$ est donnée par la formule suivante : $\sum_{L \in r'} \sum_{0 \leq i \leq size(L) - 1} card(L > L[i])$. A nouveau, du fait que les listes sont triées, la complexité de ce calcul est linéaire.

(iii) Calcul de la valeur de $S_1[3]$ et $S_1[4]$ induite par $r \in R$. On sait que $S_1[3]$ (respectivement $S_1[4]$) est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x est avant y dans le consensus et x est présent et y absent (respectivement x est absent et y absent) dans r. On sait que chaque entier k de chaque liste L de r' est associé à un élément $x \in U$ présent dans r. Pour rappel, k est l'indice du *bucket* contenant x dans le classement consensuel. Comme le nombre d'éléments qui sont manquants dans r et placés après x dans le consensus est donné par $T_2[k]$, nous en déduisons que la valeur de $S_1[3]$ induite par $r \in R$ est donnée par la formule suivante : $\sum_{L \in r'} \sum_{k \in L} T_2[k]$. Par un raisonnement analogue, nous obtenons que la valeur de $S_1[4]$ induite par $r \in R$ est donnée par la formule suivante : $\sum_{L \in r'} \sum_{k \in L} T_1[k]$. Ce calcul est de complexité linéaire.

(iv) Calcul de la valeur de $S_1[5]$ induite par $r \in R$. On sait que $S_1[5]$ est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x est avant y dans le consensus et x et y sont absents de r. Les paires à considérer sont donc nécessairement dans des *buckets* distincts du classement consensuel. Pour rappel, pour tout i, $T_3[i]$ donne le nombre d'éléments dans le *bucket* i du classement consensuel qui ne sont pas dans r. En conséquence, la valeur de $S_1[5]$ induite par $r \in R$ est donnée par la formule suivante : $\sum_{0 \leq i \leq b-1} (T_3[i] * \sum_{i+1 \leq j \leq b-1} T_3[j])$. Ce calcul est de complexité linéaire.

(v) Calcul de la valeur de $S_2[5]$ induite par $r \in R$. On sait que $S_2[5]$ est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x et y sont à égalité dans le consensus et x et y sont absents

de r. Pour rappel, pour tout i, $T_3[i]$ donne le nombre d'éléments dans le *bucket* i du classement consensuel qui ne sont pas dans r. Il faut donc sommer le nombre de paires de k éléments pour tous les k appartenant à $T_3[i]$. En conséquence, la valeur de $S_2[5]$ induite par $r \in R$ est donnée par la formule suivante : $\sum_{0 \leq i \leq b-1} \frac{T_3[i]*(T_3[i]-1)}{2}$. Ce calcul est de complexité linéaire.

(vi) Calcul de la valeur de $S_2[3] + S_2[4]$ induite par $r \in R$. On sait que $S_2[3] + S_2[4]$ est la somme pour chaque classement $r \in R$ du nombre de paires d'éléments (x, y) tels que x et y sont à égalité dans le consensus et il y a exactement un présent dans r parmi $\{x, y\}$. Pour rappel, pour tout i, $T_3[i]$ donne le nombre d'éléments dans le *bucket* i du classement consensuel qui ne sont pas dans r. Il s'agit ici de sommer pour chaque *bucket* du classement consensuel le produit du nombre d'éléments présents et du nombre d'éléments absents dans r. On appelle $size(c_i)$ la taille du *bucket* à l'indice i dans le classement consensuel. La valeur de $S_2[3] + S_2[4]$ induite par $r \in R$ est alors donnée par la formule suivante : $\sum_{0 \leq i \leq b-1} (size(c_i) - T_3[i]) * T_3[i]$. Ce calcul est de complexité linéaire.

Le code ci-dessous est l'implémentation en python de l'algorithme de calcul de $S^{(B,T)}$ présenté dans cette annexe. Il s'agit d'un extrait du code source du logiciel CoRankCo présenté dans la section 4.5.

```
from typing import Dict, List, Set
from numpy import zeros, vdot, ndarray, sort, asarray, cumsum, concatenate


class KemenyComputation:
    def __init__(self, b_vector: ndarray or List, t_vector: ndarray or List):
        self.__b_vector = asarray(b_vector)
        self.__t_vector = asarray(t_vector)

    def kemeny_score(self, consensus: List[List or Set[int or str]],
                     rankings: List[List or Set[List or Set[int or str]]]) -> float:

        # hashmap : keys = elements of dom(consensus) and values = bucket id of associated element
        mapping_elem_consensus_id_bucket = {}
        id_bucket = 0

        for bucket_consensus in consensus:
```

```
            for elem_consensus in bucket_consensus:
                mapping_elem_consensus_id_bucket[elem_consensus] = id_bucket
            id_bucket += 1

        # initialisation of sums s_1 and s_2.
        """
         s_1 (resp. s_2) must contain the number of pairs of elements (x,y) such that x < y
         (resp. x is tied with y) in the consensus and
         rank 0: x < y in input rankings
         rank 1: x > y in input rankings
         rank 2: x is tied with y in input rankings
         rank 3: x is present and y is missing in input rankings
         rank 4: x is missing and y is present in input rankings
         rank 5: x and y are missing in input rankings
        """
        s_1 = zeros(6, dtype=int)
        s_2 = zeros(6, dtype=int)

        # the cost induced by each ranking
        for ranking in rankings:
            cost_ranking = self.__cost_by_ranking(
                                        consensus, mapping_elem_consensus_id_bucket, ranking
            )
            s_1 += cost_ranking[0]
            s_2 += cost_ranking[1]
        return vdot(s_1, self.__b_vector) + vdot(s_2, self.__t_vector)

    @staticmethod
    def __cost_by_ranking(consensus: List[List or Set[int or str]],
                          mapping_elem_consensus_id_bucket: Dict[int or str, int],
                          r: List[List or Set[int or str]]) -> tuple:

        nb_buckets_consensus = len(consensus)
        s_1 = zeros(6, dtype=int)
```

```
s_2 = zeros(6, dtype=int)

# t_3[i] is the number of elements in consensus[i] missing in input ranking r
t_3 = zeros(nb_buckets_consensus, dtype=int)

missing_elements = set(mapping_elem_consensus_id_bucket.keys())
domain_ranking = set()
# r_prime is the following list of arrays : each element of each bucket of r becomes the
# number of the bucket where this element is in the consensus ranking (mapping dict)
r_prime = []

for bucket in r:
    bucket_mapped = []

    for element in bucket:
        bucket_mapped.append(mapping_elem_consensus_id_bucket.get(element))
        missing_elements.remove(element)
        domain_ranking.add(element)
    # the arrays must be sorted for the merging procedure
    r_prime.append(sort(asarray(bucket_mapped), kind='mergesort'))

# getting values of t_1, t_2 and t_3.
# t_1[i] is the number of elements missing in r and before the bucket i in the consensus
# t_2[i] is the number of elements missing in r and after the bucket i in the consensus
for element in missing_elements:
    id_bucket_consensus = mapping_elem_consensus_id_bucket[element]
    t_3[id_bucket_consensus] += 1

# t_1 is the cumulated sum of (t_3 with a 0 added as first element and without the
# last element)
t_1 = cumsum(concatenate((zeros(1), t_3[:-1])))
# t_2[i] = nb of missing elements in r - sum on j < i (t_3[j])
t_2 = len(missing_elements) - cumsum(t_3)
```

```
    """
    computation of s_1[3]
    To get the number of pairs of elements (x,y) such that x < y in consensus and x is
    present and y missing in r, we use t_2 : each integer k of r_prime is the bucket id
    of a present element x in r. As the number of missing elements of r placed
    after x in consensus is t_2[k], we increment the value of s_1[3] by t_2[k] for each
    k of r_prime.

    computation of s_1[4]
    To get the number of pairs of elements (x,y) such that x < y in consensus and x is
    missing and y present in r, we use t_1 : each integer k of r_prime is the bucket
    id of a present element x in r. As the number of missing elements of ranking
    placed before x in consensus is t_1[k], we increment the value of s_1[4] by t_1[k]
    for each k of r_prime.
    """
    for bucket in r_prime:
        for elem_r_prime in bucket:
            s_1[3] += t_2[elem_r_prime]
            s_1[4] += t_1[elem_r_prime]

    nb_missing_remaining = len(missing_elements)

    """
    computation of s_1[2]
    To get the number of pairs of elements (x,y) such that x < y in consensus and x is tied
    with y in r, we use r_prime. The value of s_1[2] is the number of distinct pairs of
    r_prime that is the sum of the product ( cardinal of each distinct value x in r_prime *
    the number of elements higher than x in r_prime).
    """

    for i in range(len(r_prime)):
        # increments s_1[2]
        s_1_2 = 0
        cursor = 0
```

```
    bucket_i_r = r_prime[i]
    bucket_size_r = len(bucket_i_r)
    while cursor < bucket_size_r - 1:
        repetition = 1
        while cursor < bucket_size_r - 1 and bucket_i_r[cursor] == bucket_i_r[cursor + 1]:
            repetition += 1
            cursor += 1
        cursor += 1
        s_1_2 += repetition * (bucket_size_r - cursor)
    s_1[2] += s_1_2

"""
computation of s_1[5] induced by ranking
To get the number of pairs of elements (x,y) such that x < y in consensus and x and y are
both missing in r, we use t_3 : t_3[i] gives the number of elements in bucket i of
consensus that are not in r. This number must be multiplied by the sum of t_3[j], j > i

computation of s_2[5] induced by ranking
To get the number of pairs of elements (x,y) such that x is tied with y in consensus and
x and y are both missing in r, we use t_3 again. This number is the sum on i of the number
of pairs of t_3[i] elements

computation of s_2[3] + s_2[4] induced by ranking (s_2[3] and s_2[4] induce same costs).
To get the number of pairs of elements (x,y) such that x is tied with y in consensus
and (x is present whereas y is missing or x is missing whereas y is present in r),
we use t_3 again. This number is the sum on i of the number of pairs (x,y) of elements
in the bucket i of consensus such that x is present and y is missing.
"""
for i in range(len(consensus)):
    # increments s_1[5], s_2[3] + s_2[4], s_2[5]
    nb_missing_elements_in_bucket_i_consensus = t_3[i]
    if nb_missing_elements_in_bucket_i_consensus > 0:
        # increments s_1[5]
        nb_missing_remaining -= nb_missing_elements_in_bucket_i_consensus
```

```
            s_1[5] += nb_missing_remaining * nb_missing_elements_in_bucket_i_consensus

            # increments s_2[3] (could be s_2[4] as well, they represent the same
            # cases / same penalty)
            s_2[3] += (len(consensus[i]) - nb_missing_elements_in_bucket_i_consensus) * \
                      nb_missing_elements_in_bucket_i_consensus

            if nb_missing_elements_in_bucket_i_consensus > 1:
                # increments s_2[5]
                s_2[5] += nb_missing_elements_in_bucket_i_consensus * (
                        nb_missing_elements_in_bucket_i_consensus - 1) / 2

    KemenyComputation.__mergesortlike(r_prime, 0, len(r_prime) - 1, s_1, s_2)

    return s_1, s_2

@staticmethod
def __mergesortlike(r_prime: List[ndarray], left: int, right: int, s_1: ndarray,
                            s_2: ndarray) -> ndarray:
    # if input ranking is empty
    if not r_prime:
        return asarray([])
    # case end of recursion
    if right <= left:
        return r_prime[right]
    # divide problem into two sub-problems of same size and merge
    else:
        middle = (right - left) // 2
        begin = middle + left + 1
        return KemenyComputation.__merge(KemenyComputation.__mergesortlike(
                                                r_prime, left, middle + left, s_1, s_2),
                                         KemenyComputation.__mergesortlike(
                                                r_prime, begin, right, s_1, s_2),
                                         s_1, s_2)
```

```
@staticmethod
def __merge(left: ndarray, right: ndarray, s_1: ndarray, s_2: ndarray):
    res = zeros(len(left) + len(right), dtype=int)
    n = len(left)
    m = len(right)
    i = 0
    j = 0
    k = 0
    while i < n and j < m:
        nb = left[i]
        nb2 = right[j]

        # no inversion
        if nb < nb2:
            res[k] = nb
            k += 1
            i += 1
        # inversion
        elif nb > nb2:
            s_1[1] += n - i
            res[k] = nb2
            k += 1
            j += 1

        # here, case where two elements were tied in the consensus and not tied in r
        else:
            cpt1 = 0
            cpt2 = 0
            while i < n and left[i] == nb:
                res[k] = nb
                k += 1
                i += 1
                cpt1 += 1
```

```
            while j < m and right[j] == nb:
                res[k] = nb
                k += 1
                j += 1
                cpt2 += 1
            # cpt1 repetitions of nb and cpt2 repetitions of nb2 with nb1 = nb2 :
            # nb1 * nb2 number of pairs of elements (x,y) such that x tied with y in
            # consensus and not tied in r
            s_2[0] += cpt1 * cpt2
            # number of added inversions
            s_1[1] += cpt2 * (n - i)

    while i < n:
        res[k] = left[i]
        k += 1
        i += 1
    while j < m:
        res[k] = right[j]
        k += 1
        j += 1

    return res
```

Bibliographie

[1] A. Agarwal, H. Raghavan, K. Subbian, P. Melville, R. D. Lawrence, D. C. Gondek, and J. Fan. Learning to rank for robust question answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 833–842, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311564. doi : 10.1145/2396761.2396867.

[2] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2) :284–300, Feb. 2010.

[3] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information : Ranking and clustering. *J. ACM*, 55(5) :23 :1–23 :27, Nov. 2008.

[4] J. A. Aledo, J. A. Gámez, and D. Molina. Approaching the rank aggregation problem by local search-based metaheuristics. *Journal of Computational and Applied Mathematics*, 354 :445 – 456, 2019. ISSN 0377-0427.

[5] A. Ali and M. Meilă. Experiments with Kemeny ranking : What works when ? *Mathematical Social Sciences*, 64 (1) :28 – 40, 2012. Computational Foundations of Social Choice.

[6] S. Amodio, A. D'Ambrosio, and 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(2) :667 – 676, 2016. ISSN 0377-2217.

[7] P. Andrieu, B. Brancotte, L. Bulteau, S. Cohen-Boulakia, A. Denise, A. Pierrot, and S. Vialette. Reliability-Aware and Graph-Based Approach for Rank Aggregation of Biological Data. In *2019 15th International Conference on eScience (eScience)*, pages 136–145, San Diego, France, Sept. 2019. IEEE.

[8] P. Andrieu, B. Brancotte, L. Bulteau, S. Cohen-Boulakia, A. Denise, A. Pierrot, and S. Vialette. Efficient, robust and effective rank aggregation for massive biological datasets. *Future Generation Computer Systems*, 2021. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2021.06.013. URL `https://www.sciencedirect.com/science/article/pii/S0167739X21002089`.

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

[10] K. J. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58(4) :328–346, 1950.

[11] I. Azzini and G. Munda. A new approach for identifying the Kemeny median ranking. *European Journal of Operational Research*, 281(2) :388 – 401, 2020. ISSN 0377-2217.

[12] G. Bachmeier, F. Brandt, C. Geist, P. Harrenstein, K. Kardel, D. Peters, and H. G. Seedig. k-majority digraphs and the hardness of voting with a constant number of voters. *Journal of Computer and System Sciences*, 105 : 130 – 157, 2019. ISSN 0022-0000.

[13] J. P. Barthelemy, A. Guenoche, and O. Hudry. Median linear orders : Heuristics and a branch and bound algorithm. *European Journal of Operational Research*, 42(3) :313–325, October 1989.

[14] J. P. Baskin and S. Krishnamurthi. Preference aggregation in group recommender systems for committee decision-making. In *Proceedings of the Third ACM Conference on Recommender Systems*, page 337–340, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584355.

[15] J. Bennett and S. Lanning. The netflix prize. *Proceedings of KDD Cup and Workshop*, Vol. 2007, 11 2006.

[16] N. Betzler and B. Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *J. Comput. Syst. Sci.*, 76 :812–836, 2010.

[17] N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. Fixed-parameter algorithms for kemeny rankings. *Theorical Computer Science*, 410(45) :4554–4570, Oct. 2009. ISSN 0304-3975. doi : $10.1016/\mathrm{j.tcs.}2009.08.033$.

[18] N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization and partial kernelization for computing medians. *Journal of Computer and System Sciences*, 77(4) :774 – 789, 2011. JCSS IEEE AINA 2009.

[19] 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*, pages 1–28, 2013.

[20] T. Biedl, F. J. Brandenburg, and X. Deng. On the complexity of crossings in permutations. *Discrete Math.*, 309 (7) :1813–1823, Apr. 2009.

[21] G. Blin, M. Crochemore, S. Hamel, S. Vialette, and B. al. Median of an odd number of permutations. *Pure Mathematics and Applications*, 21 :161 – 175, 09 2011.

[22] O. Bodenreider. The unified medical language system (umls) : integrating biomedical terminology. *Nucleic Acids Research*, 32 :D267–D270, 01 2004.

[23] B. Brancotte. *Rank aggregation with ties : algorithms, user guidance et applications to biologicals data*. Theses, Université Paris Sud - Paris XI, Sept. 2015.

[24] 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*, pages 128–142. Springer, 2014.

[25] 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)*, 8(11) :2051, Aug 2015.

[26] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, USA, 1st edition, 2016. ISBN 1107060435.

[27] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank : From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, volume 227, pages 129–136, 01 2007. doi : 10.1145/1273496.1273513.

[28] L. Ceze, J. Nivala, and K. Strauss. Molecular digital data storage using dna. *Nature Reviews Genetics*, 20 : 456–466, 2019.

[29] T. M. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, page 161–173, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 9780898716986.

[30] I. Charon and O. Hudry. A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments. *Discrete Applied Mathematics*, 154(15) :2097–2116, 10 2006. doi : 10.1016/j.dam.2005.04.020.

[31] S. Cohen-Boulakia and P. Valduriez. Interrogation et gestion de données bio-informatiques pour la biologie moléculaire. *Techniques de l'Ingénieur*, TIP140WEB :BIO7055, Nov. 2015.

[32] S. Cohen-Boulakia, A. Denise, and S. Hamel. Using medians to generate consensus rankings for biological data. In *Scientific and Statistical Database Management*, volume 6809, pages 73–90, 07 2011.

[33] J. Colomer. Ramon llull : From ars electionis to social choice theory. *Social Choice and Welfare*, 01 2012. doi : 10.1007/s00355-011-0598-2.

[34] N. d. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2014. doi : 10.1017/CBO9781139923972.

[35] V. Conitzer, A. Davenport, and J. Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, page 620–626. AAAI Press, 2006. ISBN 9781577352815.

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

[37] A. D'Ambrosio, S. Amodio, and C. Iorio. Two algorithms for finding optimal solutions of the kemeny rank aggregation problem for full rankings. *Electronic Journal of Applied Statistical Analysis*, 8 :198–213, 2015.

[38] A. Davenport and J. Kalagnanam. A computational study of the Kemeny rule for preference aggregation. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, page 697–702. AAAI Press, 2004. ISBN 0262511835.

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

[40] T. Denecker, W. Durand, J. Maupetit, C. Hébert, J.-M. Camadro, P. Poulain, and G. Lelandais. Pixel : a content management platform for quantitative omics data. *PEERJ*, 7 :e6623, 2019. doi : $10.7717/\mathrm{peerj}.6623$.

[41] T. Denecker, Y. Zhou Li, C. Fairhead, K. Budin, J.-M. Camadro, M. Bolotin-Fukuhara, A. Angoulvant, and G. Lelandais. Functional networks of co-expressed genes to explore iron homeostasis processes in the pathogenic yeast Candida glabrata. *NAR Genomics and Bioinformatics*, 2(2), June 2020. doi : $10.1093/\mathrm{nargab}/\mathrm{lqaa}027$.

[42] I. C. Dourado, D. C. G. Pedronette, and R. da Silva Torres. Unsupervised graph-based rank aggregation for improved retrieval. *Information Processing and Management*, 56(4) :1260–1279, 2019. ISSN 0306-4573.

[43] A. Duarte and R. Martí. Tabu search and grasp for the maximum diversity problem. *European Journal of Operational Research*, 178(1) :71 – 84, 2007. ISSN 0377-2217. doi : $\mathrm{https://doi.org/10.1016/j.ejor.2006.01.021}$.

[44] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. of the WWW Conference*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[45] E. J. Emond and D. W. Mason. A new rank correlation coefficient with application to the consensus ranking problem. *Journal of Multi-Criteria Decision Analysis*, 11(1) :17–28, 2002.

[46] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 301–312. ACM, 2003.

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

[48] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing partial rankings. *SIAM J. Discret. Math.*, 20(3) :628–648, Mar. 2006. ISSN 0895-4801.

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

[50] R. Genuer, J.-M. Poggi, C. Tuleau-Malot, and N. Villa-Vialaneix. Random forests for big data. *Big Data Research*, 9 :28–46, 2017. ISSN 2214-5796. doi : https://doi.org/10.1016/j.bdr.2017.07.003.

[51] P. Hall and H. Miller. Using the bootstrap to quantify the authority of an empirical ranking. *The Annals of Statistics*, 37(6B) :3929–3959, 2009. ISSN 00905364, 21688966.

[52] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, and V. A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, 33 :D514–D517, 01 2005.

[53] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of kemeny elections. *Theoretical Computer Science*, 349 :382–391, 12 2005. doi : 10.1016/j.tcs.2005.08.031.

[54] H. Ishwaran and M. Lu. Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival. *Statistics in Medicine*, 38(4) :558–582, 2019. doi : https://doi.org/10.1002/sim.7803.

[55] M. Jacob, B. Kimelfeld, and J. Stoyanovich. A system for management and analysis of preference data. *Proc. VLDB Endow.*, 7(12) :1255–1258, Aug. 2014. ISSN 2150-8097. doi : 10.14778/2732977.2732998.

[56] Y. Jiao, A. Korba, and E. Sibony. Controlling the distance to a kemeny consensus without computing it. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2971–2980. JMLR.org, 2016.

[57] A. B. Kahn. Topological sorting of large networks. *Commun. ACM*, 5 :558–562, 1962.

[58] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In O. Cheong, K.-Y. Chwa, and K. Park, editors, *Algorithms and Computation*, pages 3–14, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-17517-6.

[59] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4) :577–591, 1959. ISSN 00115266.

[60] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, page 95–103, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936318.

[61] R. Kolde, S. Laur, P. Adler, and J. Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics (Oxford, England)*, 28 :573–80, 02 2012.

[62] C. Kuhlman and E. Rundensteiner. Rank aggregation algorithms for fair consensus. *Proc. VLDB Endow.*, 13 (12) :2706–2719, July 2020. ISSN 2150-8097. doi : 10.14778/3407790.3407855.

[63] H. Kujawska, M. Slavkovik, and J.-J. Rückmann. Predicting the winners of borda, kemeny and dodgson elections with supervised machine learning. In N. Bassiliades, G. Chalkiadakis, and D. de Jonge, editors, *Multi-Agent Systems and Agreement Technologies*, pages 440–458, Cham, 2020. Springer International Publishing. ISBN 978-3-030-66412-1.

[64] X. Li, X. Wang, and G. Xiao. A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications. *Briefings in Bioinformatics*, 20(1) :178–189, 08 2017. doi : $10.1093/\mathrm{bib}/\mathrm{bbx101}$.

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

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

[67] M. Meilă, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'07, page 285–294, Arlington, Virginia, USA, 2007. AUAI Press. ISBN 0974903930.

[68] R. Milosz. *Étude algorithmique et combinatoire de la méthode de Kemeny-Young et du consensus de classements*. Theses, Université de Montréal, 2019.

[69] R. Milosz and S. Hamel. Heuristic, branch-and-bound solver and improved space reduction for the median of permutations problem. In L. Brankovic, J. Ryan, and W. F. Smyth, editors, *Combinatorial Algorithms*, pages 299–311, Cham, 2018. Springer International Publishing.

[70] R. Milosz, S. Hamel, and A. Pierrot. Median of 3 permutations, 3-cycles and 3-hitting set problem. In C. Iliopoulos, H. W. Leong, and W.-K. Sung, editors, *Combinatorial Algorithms*, pages 224–236, Cham, 2018. Springer International Publishing. ISBN 978-3-319-94667-2.

[71] E. Moreno-Centeno and A. R. Escobedo. Axiomatic aggregation of incomplete rankings. *IIE Transactions*, 48 (6) :475–488, 2016.

[72] R. Morris and B. J. Truskowski. The evolution of storage systems. *IBM Syst. J.*, 42 :205–217, 2003.

[73] S. Muravyov and I. Marinushkina. Intransitivity in multiple solutions of Kemeny ranking problem. *Journal of Physics Conference Series*, 459 :012006, 09 2013.

[74] S. Muravyov, P. Baranov, and E. Emelyanova. How to transform all multiple solutions of the kemeny ranking problem into a single solution. *Journal of Physics : Conference Series*, 1379 :012053, 11 2019. doi : $10.1088/1742\text{-}6596/1379/1/012053$.

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

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

[77] D. J. Rigden and X. M. Fernández. The 2021 Nucleic Acids Research database issue and the online molecular biology database collection. *Nucleic Acids Research*, 49(D1) :D1–D9, 12 2020. ISSN 0305-1048. doi : 10.1093/nar/gkaa1216.

[78] F. Schalekamp and A. van Zuylen. Rank aggregation : Together we're strong. In *Proc of ALENEX*, pages 38–51, 2009.

[79] M. Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2) :267–303, 2011.

[80] Z.-y. Song, F. Chao, Z. Zhuo, Z. Ma, W. Li, and G. Chen. Identification of hub genes in prostate cancer using robust rank aggregation and weighted gene co-expression network analysis. *Aging*, 11, 07 2019. doi : 10.18632/aging.102087.

[81] M. Stearns, C. Price, K. Spackman, and A. Wang. Snomed clinical terms : Overview of the development process and project status. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*, pages 662–6, 02 2001.

[82] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2), 1972.

[83] 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.

[84] Y. Yoo, A. R. Escobedo, and J. K. Skolfield. A new correlation coefficient for comparing and aggregating non-strict and incomplete rankings. *Eur. J. Oper. Res.*, 285 :1025–1041, 2020.

[85] H. P. Young. *Equity : In Theory and Practice*. Princeton University Press, 1995.

[86] H. P. Young and A. Levenglick. A consistent extension of condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2) :285–300, 1978. ISSN 00361399.

[87] V. Švendová and M. G. Schimek. A novel method for estimating the common signals for consensus across multiple ranked lists. *Computational Statistics and Data Analysis*, 115 :122–135, 2017. ISSN 0167-9473.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Passage à l'échelle, propriétés et qualité des algorithmes de classements consensuels pour les données biologiques massives

Mots clés : méthode Kemeny-Young, agrégation de classements, données biologiques, classements incomplets, algorithmique, théorie du choix social

Résumé :
Les médecins et biologistes sont régulièrement amenés à interroger des bases de données biologiques publiques, par exemple lorsqu'ils se renseignent sur les gènes les plus associés à une maladie donnée. Le mot-clé choisi au moment d'interroger la base de données est particulièrement important : plusieurs reformulations synonymes d'une même maladie (par exemple « breast cancer » et « breast carcinoma ») aboutissent à des classements parfois très différents pouvant aller jusqu'à plusieurs milliers de gènes. Certains gènes, triés par pertinence, peuvent se retrouver à égalité (importance égale vis-à-vis de la maladie). De plus, certains gènes retournés en utilisant certaines reformulations peuvent être absents lorsque d'autres reformulations sont choisies. On dit alors que les classements sont incomplets et avec égalités. L'enjeu est alors de combiner l'information apportée par ces différents classements de gènes. La problématique consistant à partir d'une liste de classements et à calculer un classement dit consensuel aussi représentatif que possible des classements d'entrée est appelée « agrégation de classements ». Ce problème est connu pour être NP-difficile. Alors que la majorité des travaux considèrent les classements complets et sans égalités, nous nous sommes placés dans le contexte des classements incomplets avec égalités. Nos contributions peuvent se décomposer en trois parties. Premièrement, nous avons conçu une heuristique basée sur des graphes qui consiste à partitionner le problème de départ en sous-problèmes indépendants pour le cas où les classements sont incomplets et avec égalités. Deuxièmement, nous avons conçu un algorithme capable de déterminer des points communs entre tous les classements consensuels optimaux, permettant ainsi de fournir à l'utilisateur une indication quant à la robustesse du classement consensuel renvoyé. Une étude expérimentale sur un grand nombre de jeux de données biologiques massifs a mis en évidence la pertinence biologique des résultats fournis par nos méthodes. La dernière contribution est la suivante : les données manquantes pouvant s'interpréter de différentes façons selon le contexte, nous avons proposé un modèle paramétré permettant de prendre en compte ces différences. Nous avons conçu des algorithmes pour ce modèle et fait une étude axiomatique de ce dernier en nous basant sur la théorie du choix social.

Title : Scalability, features and quality aspects of consensus ranking algorithms for big biological data sets

Keywords : Kemeny-Young method, rank aggregation, biological data, incomplete rankings, algorithmics, social choice theory

Abstract : Biologists and physicians regularly query public biological databases, for example when they are looking for the most associated genes towards a given disease. The chosen keyword are particularly important : synonymous reformulations of the same disease (for example "breast cancer" and "breast carcinoma") may lead to very different rankings of (thousands of) genes. The genes, sorted by relevance, can be tied (equal importance towards the disease). Additionally, some genes returned when using a first synonym may be absent when using another synonym. The rankings are then called "incomplete rankings with ties". The challenge is to combine the information provided by these different rankings of genes. The problem of taking as input a list of rankings and returning as output a so-called consensus ranking, as close as possible to the input rankings, is called the "rank aggregation problem". This problem is known to be NP-hard. Whereas most works focus on complete rankings without ties, we considered incomplete rankings with ties. Our contributions are divided into three parts. First, we have designed a graph-based heuristic able to divide the initial problem into independent sub-problems in the context of incomplete rankings with ties. Second, we have designed an algorithm able to identify common points between all the optimal consensus rankings, allowing to provide information about the robustness of the provided consensus ranking. An experimental study on a huge number of massive biological datasets has highlighted the biological relevance of these approaches. Our last contribution the following one : we have designed a parameterized model able to consider various interpretations of missing data. We also designed several algorithms for this model and did an axiomatic study of this model, based on social choice theory.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France