Université de Montréal

Étude de la médiane de permutations sous la distance de Kendall-Tau

par
Robin Milosz

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

décembre, 2015

RÉSUMÉ

La distance de Kendall-τ compte le nombre de paires en désaccord entre deux permutations. La distance d'une permutation à un ensemble est simplement la somme des distances entre cette permutation et les permutations de l'ensemble. À partir d'un ensemble donné de permutations, notre but est de trouver la permutation, appelée médiane, qui minimise cette distance à l'ensemble.

Le *problème de la médiane de permutations sous la distance de Kendall-τ*, trouve son application en bio-informatique, en science politique, en télécommunication et en optimisation.

Ce problème d'apparence simple est prouvé difficile à résoudre. Dans ce mémoire, nous présentons plusieurs approches pour résoudre le problème, pour trouver une bonne solution approximative, pour le séparer en classes caractéristiques, pour mieux comprendre sa compléxité, pour réduire l'espace de recheche et pour accélérer les calculs. Nous présentons aussi, vers la fin du mémoire, une généralisation de ce problème et nous l'étudions avec ces mêmes approches.

La majorité du travail de ce mémoire se situe dans les trois articles qui le composent et est complémenté par deux chapitres servant à les lier.

Mots clés: algorithmes, bio-informatique, système de classements, heuristiques, optimisation combinatoire, médiane, permutation, contraintes.

ABSTRACT

The Kendall-τ distance counts the number of pairwise disagreements between two permutations. The distance between a permutation and a set is simply the sum of the distances between the considered permutation and the permutations of the set. Given a set of permutations, we want to find the permutation, called median, that minimise that distance to the set.

The *problem of finding a median of permutations under the Kendall-τ distance*, finds applications in bioinformatics, political science, telecommunications and optimization.

This simple appearing problem is proven difficult to solve. In this master thesis, we present a few approaches to solve the problem, to find a good approximate solution, to separate it into caracteristic classes, to deepen our understanding of its complexity, to reduce the search space and to accelerate calculations. We also present, at the end of this thesis, a generalization of this problem and we study it with the same approaches.

The majority of the work in this thesis is located in the three papers which compose it and is complemented by two chapters, that bound them all together.

Keywords: algorithms, bio-informatic, ranking systems, heuristic, combinatorial optimization, median, permutation, constraints.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES

B&B	Branch and Bound
KT	Kendall-Tau
MOT	Major Order Theorem
SA	Simulated Annealing

NOTATION

π, σ	Permutations
n	Taille des permutations
S_n	Espace des permutations
$a \prec b$	Élément a préféré à l'élément b
$d_{KT}(\pi, \sigma)$	Distance de Kendall-τ
A	Ensemble de permutations de départ
m	Nombre de permutations de départ, taille de A
π^*	Permutation médiane
$M(A)$	Ensemble des permutations médianes
L	Matrice gauche
R	Matrice droite
R	Classement
$\mathscr{R}$	Ensemble de classements de départ
$Rank_n$	Espace des classements
$a = b$	Élément a à égalité avec l'élément b
E	Matrice égalité (pour le chapitre 5)
$\mathbb{S}^2$	Sphère

À la science,

REMERCIEMENTS

J'aimerais remercier toutes les personnes qui m'ont aidé au cours de ma maîtrise :
Ma directrice de recherche Sylvie Hamel qui m'a guidé dans cette maîtrise et offert tellement d'opportunités pour avoir une magnifique expérience académique.
Ma famille : Anna Klepacka, ma mère, qui m'a soutenu et qui a été mon oracle de sagesse, Julien Milosz, mon frère, qui m'a donné une appréciation encore plus grande de la science et Nicole Burke, mon amour, qui m'a encouragé et qui m'a aidé dans la révision de mon article en anglais.
Mes vélos (le courant "Wasp" et le défunt "Bumblebee"), reconnaissable par leurs motifs jaune et noir, qui m'ont fidèlement transporté tous les jours jusqu'à l'université, autant les jours motivants que les jours où la fatigue se sentait fort, dans toutes les conditions allant du beau soleil jusqu'à la pluie, des canicules d'été jusqu'au froids horribles de l'hiver, de la belle chaussée dégagée à la rue congestionnée de trafic à cause d'une tempête de neige. Ils m'ont aussi permis de me calmer avant de m'asseoir pour travailler au lab.
Les stagiares : Vincent Antaki et Charles Desharnais, qui pendant l'été 2015 ont travaillé sur le même problème et ont fait avancer la recherche dans différents volets grâce à leurs forces en informatique et en mathématique.
Les collaborateurs de France : Bryan Bancotte, Sarah Cohen-Boulakia et Alain Denise qui m'ont donné l'opportunité de travailler avec eux en France et qui m'ont inspiré dans la découverte la plus belle de ma maîtrise.
La responsable des laboratoires de bio-info : Marie Pageau qui m'a offert des superbes expériences d'enseignement.
Une mention pour Manuel Lafond, collègue de lab présentement au Ph.D., qui a toujours été là depuis mon entrée dans le lab, avec qui j'ai pu échanger une quantité indénombrable de fois sur les sujets de la recherche et de la vie académique.
Finalement, un remerciement au FQRNT pour le soutien financier qui m'a permis de faire cette maîtrise.

AVANT-PROPOS

La recherche montrée dans ce travail s'inscrit dans une longue collaboration avec ma directrice Sylvie Hamel, avec qui je travaille depuis l'été 2012 à l'Université de Montréal :

Stage d'été 2012 avec une BRCP du CRSNG,
Projet final du cours IFT6291 à l'automne 2012,
Cours-projet d'informatique IFT3150 à hiver 2013,
Stage d'été 2013 avec une BRCP du CRSNG,
Et présentement, au moment de la rédaction, en maîtrise en informatique ! (hiver 2014 - automne 2015)

Donc je ne suis pas parti de zéro au moment de commencer ma recherche, plusieurs idées trottaient déjà dans ma tête.

Au cours de ma maîtrise, la recherche a donné le matériel qui a permis à la rédaction de 3 articles dont un publié, un accepté et un refusé, mais en réécriture.

INTRODUCTION

Présentation du problème

Le problème de la médiane d'un ensemble de m permutations des éléments $\{1,2,\ldots n\}$ sous la distance de Kendall-τ [21] [37] est souvent cité dans la littérature comme le Kemeny Score Problem [20]. Dans ce problème, m électeurs vont individuellement ordonner une liste de n candidats selon leurs préférences. Le problème consiste à trouver un consensus de Kemeny : un ordre des candidats qui s'accorde le mieux avec les ordres des m électeurs c'est-à-dire qui minimise la somme des désaccords. Le problème est trivial pour $m = 1$, résolvable en temps polynomial pour $m = 2$, a été montré NP-Complet quand $m \geq 4$, m pair (en premier dans [16], puis corrigé dans [7]) mais la complexité demeure inconnue pour $m \geq 3$, m impair. Dans les 10 dernières années, plusieurs algorithmes d'approximations ont été proposés ; premièrement, un algorithme probabiliste avec un facteur d'approximation $11/7$ [2] puis ensuite un algorithme déterministe avec un facteur d'approximation $8/5$ [38]. En 2007, un schéma d'approximation en temps polynomial (PTAS) a été obtenu dans [22] et quelques années plus tard, plusieurs algorithmes à paramètres fixes (fixed parameter algorithms) ont été décrits [5] [19] [32] [35]. Une approche théorique réduisant l'espace de recherche a aussi été proposée dans [6].

Applications

Le problème de la médiane de permutations s'applique à plusieurs problèmes dans divers domaines des sciences. Pour ne donner que quelques exemples, il apparaît en bio-informatique [34] dans le classement de protéines/gènes liées à une maladie [10], la construction de cartes génétiques [14] et la phylogénétique.

En science politique, il permet l'ordonnancement des priorités de sujet à traiter, de faire des élections utilisant le consensus de Kemeny et le classement de choix sociaux et de compétiteurs sportifs [6].

En télécommunications, plus anciennement, il a permis la construction d'engin de meta-recherche web et la détection de spam [16].

Finalement, le problème entre en jeu dans n'importe quel système où l'on effectue plusieurs classements et où on s'intéresse à un consensus. On peut reconstruire une suite d'événements à partir d'histoires racontées. On peut même ordonner les films et émissions de télévision sur le site d'un fournisseur selon les classements personnalisés de chaque utilisateur [36].

Plan du mémoire

Le travail du mémoire se décompose en cinq volets chacun représenté par un chapitre. Le côté pratique et le côté théorique ont été parallèlement investigués sachant que la dualité bénéficie un à l'autre. Le travail est encadré par une introduction et une conclusion.

Le chapitre 1 introduit le problème et son contexte. On y introduit aussi les définitions nécessaires à la compréhension de ce mémoire.

Le chapitre 2, un volet pratique, fait le tour de différentes méthodes de résolutions du problème à l'aide de différentes approches algorithmiques.

Le chapitre 3, un volet mi-pratique\mi-théorique soutenu par un article publié, présente des contraintes qui réduisent l'espace de recherche du problème.

Le chapitre 4, un volet théorique soutenu par un "extended abstract" publié, investigue une sous-classe du problème qui peut être résolue en temps polynomial.

Le chapitre 5, un volet mi-pratique\mi-théorique qui s'appuie sur un article non publié, offre une généralisation du problème aux ensembles de permutations avec égalités.

Finalement, la conclusion fait un rappel des lignes importantes du travail et offre des voies pour de futurs développements.

CHAPITRE 1

CONTEXTE ET DÉFINITIONS

1.1 Contexte

Le problème de la médiane de permutations consiste à trouver un ordre consensus à partir de plusieurs ordres qui minimise le nombre de désaccords entre les éléments ordonnés. (Voir Figure 1.1)

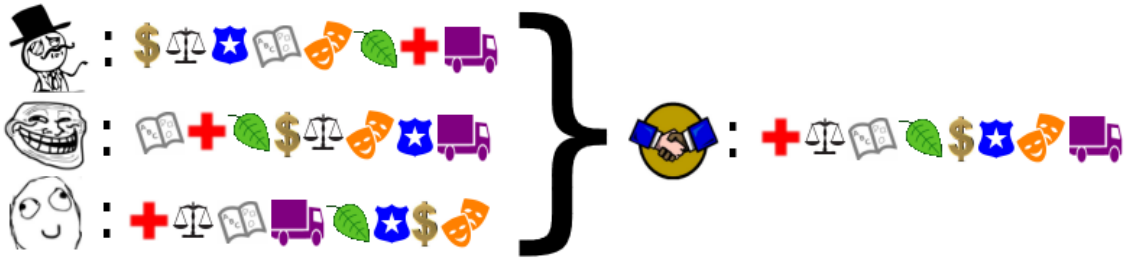


Figure 1.1 – Exemple d'introduction : trois citoyens ordonnent les sujets politiques selon leurs importances personnelles pour un éventuel débat ou pour une prise de décision. Sachant que les opinions sont différentes, on utilise le consensus de Kemeny pour trouver l'ordre consensus qui respectera le plus possible les ordres individuels. Dans cet exemple, il conviendra de traiter ces sujets dans l'ordre : "santé, justice, éducation, environnement, économie, sécurité, culture et transport". Nous allons utiliser cet exemple au cours du mémoire et la modélisation de ce problème se trouve à la Figure 2.6.

Le problème est NP-Complet, lorsque le nombre d'ordres pour lesquels on cherche un consensus est pair et ≥ 4 [16]. Ce qui signifie qu'il n'y a pas de moyen facile de trouver le consensus dans n'importe quel cas. Il est alors pertinent de chercher des astuces pour aider à résoudre ce problème, soit pour améliorer la vitesse des algorithmes exacts ou pour développer des heuristiques efficaces.

1.2 Définitions

Il faut définir les objets mathématiques que nous allons utiliser : Une **permutation** π de n est une bijection de l'ensemble $\{1,2,\ldots,n\}$ sur lui-même, c'est donc un ordre

total des éléments $\{1,2,\ldots,n\}$. On dénote une permutation π de $\{1,2,\ldots,n\}$ par $\pi = \pi_1\pi_2\ldots\pi_n$ et $\pi[j]$ dénote la position de l'entier j dans π. (voir Figure 1.2)

$$\pi = [\ 3,\ 5,\ 1,\ 7,\ 8,\ 6,\ 2,\ 4]$$

Figure 1.2 – π une permutation de $n = 8$. Dans cette permutation, l'élément 5 se retrouve avant l'élément 1. On note alors que l'ordre entre 5 et 1 est $5 \prec 1$. On note $\pi[7] = 4$ pour dire que l'élément 7 se trouve à la position 4.

Tout au long de ce mémoire, A représentera un **ensemble de permutations**. On peut aussi appeler A une collection de permutations, car on permet d'avoir des permutations dupliquées dans l'ensemble. Cette propriété vient en réponse à l'application réelle où rien n'empêche deux compétitions d'avoir un résultat identique. La cardinalité de A est le nombre m de permutations qui s'y trouvent, dénotée $\#A = m$. Toutes les permutations d'un certain ensemble de permutations sont de même taille et contiennent les mêmes éléments.

L'**espace des permutations** S_n est l'ensemble contenant toutes les permutations de taille n (voir Figure 1.3 pour un exemple visuel). C'est notre espace de recherche.

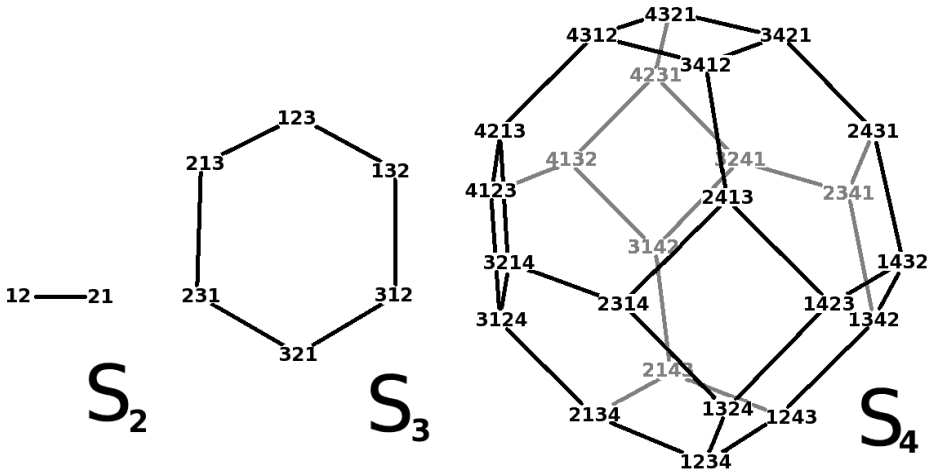


Figure 1.3 – L'espace des permutations, exemples pour S_2, S_3 et S_4.

On définit l'**ordre entre deux éléments** d'une permutation comme suit : $a \prec b$ signifie que a est avant b ou que a est préféré à b alors que $b \prec a$, l'inverse, signifie que b est avant a ou b est préféré à a. Dans une permutation, il n'y a qu'un seul ordre possible

pour une paire d'éléments. Pour traiter les ensembles de permutations, on utilise deux matrices : L la **matrice "gauche"** et R la **matrice "droite"** dont la case $L[i][j]$ (resp. $R[i][j]$) dénote le nombre de permutations de l'ensemble où l'élément i est à gauche de l'élément j, $i < j$ (resp. i à droite de j, $j < i$). Notez que $L[i][j] = m - R[i][j]$.

La **distance de Kendall-τ** se définit entre deux permutations par le nombre de paires d'éléments qui ne sont pas dans le même ordre, autrement dit ; le nombre de paires d'éléments en désaccord :

$$d_{KT}(\pi,\sigma) = \#\{(i,j) | i < j \wedge (\pi[i] < \pi[j] \wedge \sigma[i] > \sigma[j]) \vee (\pi[i] > \pi[j] \wedge \sigma[i] < \sigma[j])\}$$

La figure 1.4 détaille un exemple de cette distance entre deux permutations. On remarque que la distance compte aussi le nombre minimal d'inversions d'éléments adjacents pour passer d'une permutation à l'autre. Il est possible alors de calculer la distance entre deux permutations de taille n en $O(nlog(n))$ en modifiant légèrement l'algorithme du tri fusion. On définit la distance d'une permutation à un ensemble comme la somme des distances entre la permutation et les permutations de l'ensemble.

$$d_{KT}(\pi,A) = \sum_{\sigma \in A} D_{KT}(\pi,\sigma)$$

Mathématiquement, le **problème de la médiane de permutations** se définit comme suit : Soit $A \subset S_n$ un ensemble de permutations. Trouver l'ensemble $M(A)$ des permutations $\pi^* \in S_n$ tel que $d_{KT}(\pi^*,A) \leq d_{KT}(\pi,A), \forall \pi \in S_n$. Une **permutation médiane** pour un ensemble A est donc une permutation qui est la plus proche de A sous cette distance. L'ensemble **M(A)** est l'ensemble de toutes les permutations médianes de A. (Voir Figure 1.5)

Dans une variante, on pourrait être intéressé seulement par l'obtention d'une ou quelque permutations médianes ou même simplement par l'obtention de la distance médiane. Dans les travaux de ce mémoire, on visera à trouver soit une permutation médiane soit l'ensemble $M(A)$ au complet.

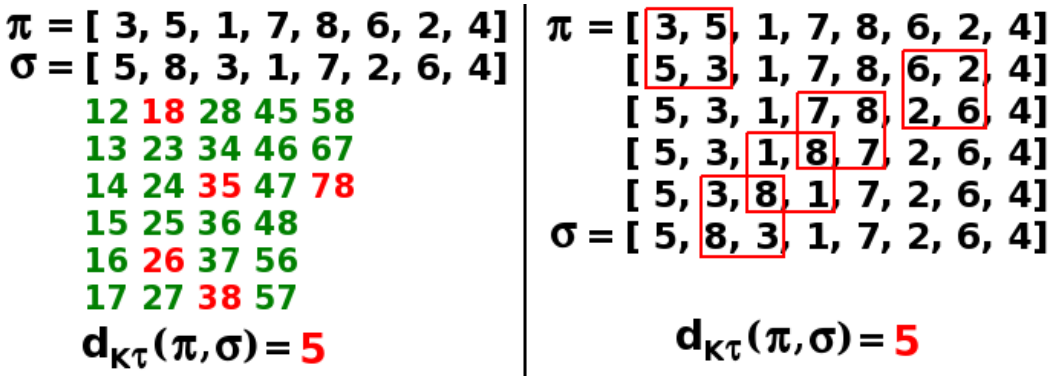


Figure 1.4 – Distance de Kentall-τ entre deux permutations π et σ. À gauche, les paires en rouges représentent les désaccords et leur nombre est la distance. C'est équivalent au nombre de swaps (interchanger deux éléments adjacents) nécessaires pour passer d'une permutation à l'autre. À droite on peut observer les swaps encadrés en rouge.

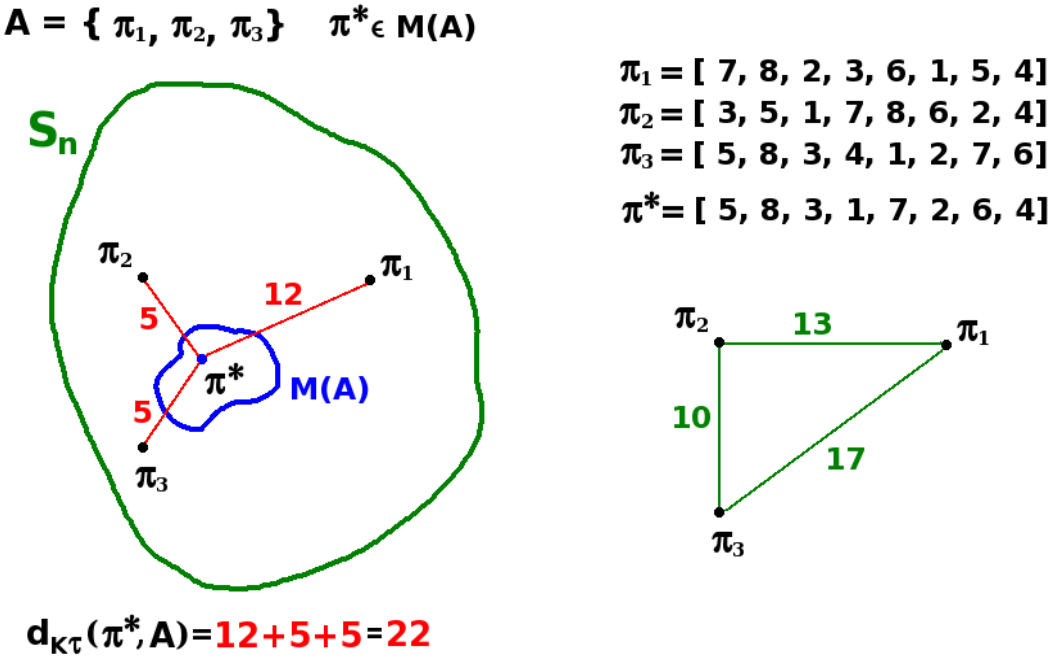


Figure 1.5 – Vue d'artiste du problème de la médiane de permutations.

CHAPITRE 2

ALGORITHMES ET HEURISTIQUES

Le problème de la médiane de permutations a plusieurs applications (voir l'introduction) alors il est intéressant de pouvoir calculer la solution au problème de manière efficace d'autant plus que le probème est difficile. L'intérêt des méthodes exactes se trouve dans un contexte politique ou de compétition où la solution optimale est nécessaire alors que l'intérêt des méthodes heuristiques se trouve dans un contexte de données telles les préférences de film ou la métarecherche (assembler les résultats de plusieurs moteurs de recherche, etc.).
Dans ce chapitre, nous présentons les approches exactes et heuristiques que nous avons testées pour résoudre le problème de la médiane de permutations, mais d'abord, nous allons présenter ce qui a déjà été fait sur ce sujet.

2.1 Travaux précédents

Différents travaux on été réalisés dans les dernières années, approchant le problème sous différents angles.

Tout d'abord des premières approximations ont été réalisées avec des algorithmes déterministes et probabilistes. En 2001, une approximation de facteur 2 en temps polynomial a été proposée dans [16], elle est basée sur les chaînes de Markov. En 2008, un algorithme probabiliste avec un facteur d'approximation $11/7$ a été proposé dans [2]. Cet algorithme est basé sur une modification de l'algorithme QuickSort et sur le choix aléatoire d'une permutation de départ (qui est en soit une approximation de facteur 2 [15]). Cet algorithme est d'un facteur $6/5$ lorsqu'il est appliqué à trois permutations en entrée. Cela a été suivi en réponse par [38] qui propose un algorithme combinatoire déterministe avec un facteur d'approximation $8/5$ et des versions déterministes des algorithmes

de [2].

Une heuristique gloutone d'un facteur d'approximation d'ordre 2 a été décrite dans [9] en 1999. En 2004, [13] propose une autre heuristique gloutone qui est plus efficace dans la majorité des cas. Ces heuristiques ont été utilisées sur des donnés allant jusqu'à $n = 50$ et $m = 25$.

Un schéma d'approximation en temps polynomial (PTAS) a été obtenu dans [22] en 2007. Cet algorithme considéré d'intérêt théorique seulement par [38], [35] et [30] mais considéré important par [2].

Pour l'approche exacte, un algorithme "Branch and Bound" a été proposé en 2004 dans [13]. L'étude se concentre sur des instances où le nombre de permutations de départ est impair et où la corrélation entre permutations de départ est forte. L'algorithme peut résoudre des instances allant jusqu'à $n = 15$. Cela a été suivi d'une amélioration de la borne inférieure en 2006 dans [11], qui utilise CPLEX pour obtenir la distance optimale des instances allant jusqu'à une taille de $n = 40$. Cet algorithme "Branch and Bound" est repris en 2007 par l'étude [28] pour obtenir des solutions exactes pour des instances de petites tailles $n = 8, 10$. Une heuristique hybride [28] utilisant également l'approche "Branch and Bound" est utilisée sur des instances de taille allant jusqu'à $n = 25$.

Une partie des études nommées ci-haut suggère un lien fort entre une mesure de concordance (similarité entre les permutations de départ) du problème et sa difficulté a être résolu.

Récemment (2008-2014), les chercheurs se sont apperçus que la similarité des permutations de l'ensemble de départ avait une grande influence sur la difficulté du problème [13] et plusieurs algorithmes à paramètres fixes (fixed parameter algorithms) ont été décrits. Premièrement, dans [30] et [31], on propose une approche basée sur une distance moyenne faible entre les permutations de départ. Cela a été davantage investigué et raffiné par [35], [19] et [32]. On y retrouve d'ailleurs des résultats sur l'énumération des solutions optimales, sous paramètres fixes. Dans [5], est introduit un premier concept de kernelization (réduction d'une instance à une instance plus petite qui conserve les mêmes propriétés) parametrée par la distance moyenne. Cette approche est davantage explorée dans [6] qui lie la kernelization avec une réduction de l'espace de recherche.

Une étude comparative a été faite en 2011 dans [3]. Cette étude compare différents algorithmes exactes ou heuristiques sur des données réelles ou synthétiques et propose une mesure de difficulté d'une instance du problème. L'étude, en conclusion recommandé d'utiliser si possible un algorithme exacte, indique que la méthode de résolution exacte la plus efficace est la programation linéaire en nombres entiers avec CPLEX et propose de complémeter les algorithmes avec des techniques de prétraitement pour réduire l'espace de recherche.

2.2 Simulated Annealing

2.2.1 Théorie

L'heuristique probabiliste du recuit simulé ou "Simulated Annealing" en anglais (SA) [23][39] se retrouve entre le hill climbing et une marche aléatoire dans l'espace. Cette méthode se base sur le principe de recuit en métallurgie. On commence avec une solution aléatoire consistante (une solution valide qui respecte les contraintes mais qui n'est pas nécessairement optimale) et une haute température. À chaque étape, une solution alternative est proposée par une perturbation de la solution courante. Si cette solution offre un score meilleur ou égal au score de la solution courante, on remplace la solution courante par cette nouvelle solution. Par contre, si le score est moins intéressant (différence de score = Δ), on choisit cette solution "moins bonne" avec une probabilité suivant la loi de Boltzman ($P = e^{\frac{-\Delta}{T}}$) (voir Figure 2.1). La température du système baisse graduellement. Plus la différence Δ est grande, plus la probabilité d'accepter la solution moins bonne proposée diminue. La baisse de la température T amène aussi une baisse de la probabilité d'accepter de mauvaises solutions. On désigne parfois la solution courante par "électron". (Voir le pseudo-code Algortithm 1, 10).

Cette méthode heuristique permet de bien explorer l'espace de recherche et sa propriété d'accepter des moins bonnes solutions avec une certaine probabilité permet éventuellement de sortir des minimums locaux. Il a été démontré théoriquement [26] que SA converge toujours vers la solution optimale si le temps accordé à la simulation tend vers l'infini. En pratique, on dispose d'un temps limité. On va alors aller chercher les para-

Algorithm 1 Simulated Annealing

Data: Fonction F à minimiser
Result: Ensemble des meilleures solutions trouvées *BestC*
begin
  $\lambda \longleftarrow$ COOLING ;
  $BestC \longleftarrow \{\}$;
  for $i \leftarrow 1$ ***to*** $nbREPETITIONS$ **do**
    $T \longleftarrow$ MAXTEMP;
    $s \longleftarrow$ solution aleatoire consistante;
    for $i \leftarrow 1$ ***to*** $MAXITE$ **do**
      $T \longleftarrow T * \lambda$;
      $s_{new} \longleftarrow$ choisirAleatoirement(voisins(s));
      $\Delta \longleftarrow F(s_{new}) - F(s)$;
      if $\Delta \leq 0$ **then**
        $s \longleftarrow s_{new}$
      else
        $rand \longleftarrow random[0,1]$;
        if $rand < e^{-\Delta/T}$ **then**
          $s \longleftarrow s_{new}$
        endif
      endif
    endfor
    $BestC \longleftarrow BestC \cup \{s\}$
  endfor
  return *BestC* ;
end

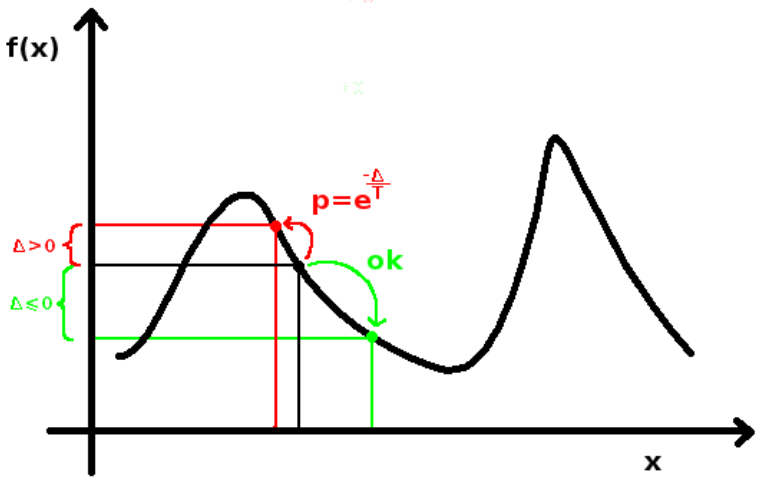


Figure 2.1 – Acceptation des mouvements par SA. Le graphe présente le panorama d'un espace de recherche, avec l'axe horizontal représentant les solutions possibles et l'axe vertical la valeur de la fonction objective pour ces solutions. On cherche à minimiser cette fonction. La solution courante est identifiée en noir. Deux solutions alternatives du voisinage sont représentées. La première solution alternative (en vert) a une valeur plus faible de la fonction objective, cette solution va être acceptée automatiquement. La deuxième solution alternative (en rouge) augmente la valeur de la fonction objective par Δ, elle sera acceptée qu'avec une probabilité de $P = e^{\frac{-\Delta}{T}}$. Notez que seulement une solution alternative est proposée à la fois.

mètres optimaux qui augmentent la probabilité d'obtenir la bonne solution en le moins de temps possible.

Dans notre problème, l'espace de recherche S_n est relativement bien adapté pour cette méthode. Premièrement, l'espace est fermé (à géométrie fermée, voir Figure 1.3) ce qui permet d'avoir plusieurs chemins alternatifs opposés qui mènent au même point (telle la surface d'une sphère, $\mathbb{S}^2$). Deuxièmement, il a généralement une forme contenant des grands bassins d'attraction contrairement à un espace plus uniforme. Ces bassins vont se situer proche des permutations de départ. Troisièmement, l'espace est dénombrable et fermé, il existe donc une quantité limité de solution à explorer.

Dans notre contexte, une solution est simplement une permutation, alors qu'une solution de départ est un permutation aléatoire générée par le mélange Fisher-Yates [17][25].

Dans le but de trouver un voisin de notre permutation courante, plusieurs mouvements de base possibles sur les permutations peuvent être considérés, parmi les plus connus : 1) **swap** - interchanger deux éléments adjacents, 2) **transposition** - interchan-

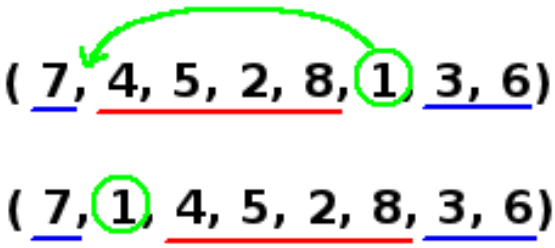


Figure 2.2 – Mouvement circulaire vers la droite dans une permutation du segment "4,5,2,8,1". Les éléments 4,5,2,8 sont déplacés d'une position vers la droite alors que l'élément 1 est renvoyé à gauche de ces éléments. Les éléments 7,3,6 ne sont pas affectés.

ger deux éléments peu importe leurs positions dans la permutation, 3) **mouvement circulaire** - vers la droite (resp. vers la gauche) d'un segment d'une permutation i.e. le déplacement d'une position vers la droite (resp. vers la gauche) de tous les éléments du segment sauf pour le dernier (resp. le premier) élément du segment qui est envoyé à l'avant (resp. à l'arrière) des autres éléments du segment (voir Figure 2.2), et 4) **inversion de bloc** - inverser l'ordre des éléments dans un intervalle.

Nous avons analysé le potentiel de chaque mouvement. Les **swaps** sont simples à calculer, mais ne permettent pas de grands mouvements. De plus, l'espace devient difficile à naviguer ; on peut imaginer n'importe quelle permutation non optimale dont toutes les paires adjacentes sont optimales, ce qui en fait un minimum local (voir Figure 2.3). Les **inversions de blocs** sont lourdes à calculer, car elles inversent beaucoup de paires et les déplacements dans l'espace sont plus grands. Par contre, ces mouvements sont moins aptes à faire des petits changements dans la solution ce qui s'avère crucial dans les dernières phases pour l'obtention de la solution optimale. En effet, on a observé que la plupart des solutions proches de l'optimum se trouvent à juste quelques mouvements circulaires près. Un mouvement circulaire pourrait être atteint par concaténation de deux mouvements d'inversion de bloc, mais la probabilité de renverser un bloc dans une solution proche de l'optimum est faible, d'autant plus si la température est faible. La **transposition** peut être vue comme une combinaison de deux mouvements circulaires. Finalement, le **mouvement circulaire** permet des déplacements moyens ou petits et se calcule dans un temps raisonnable.

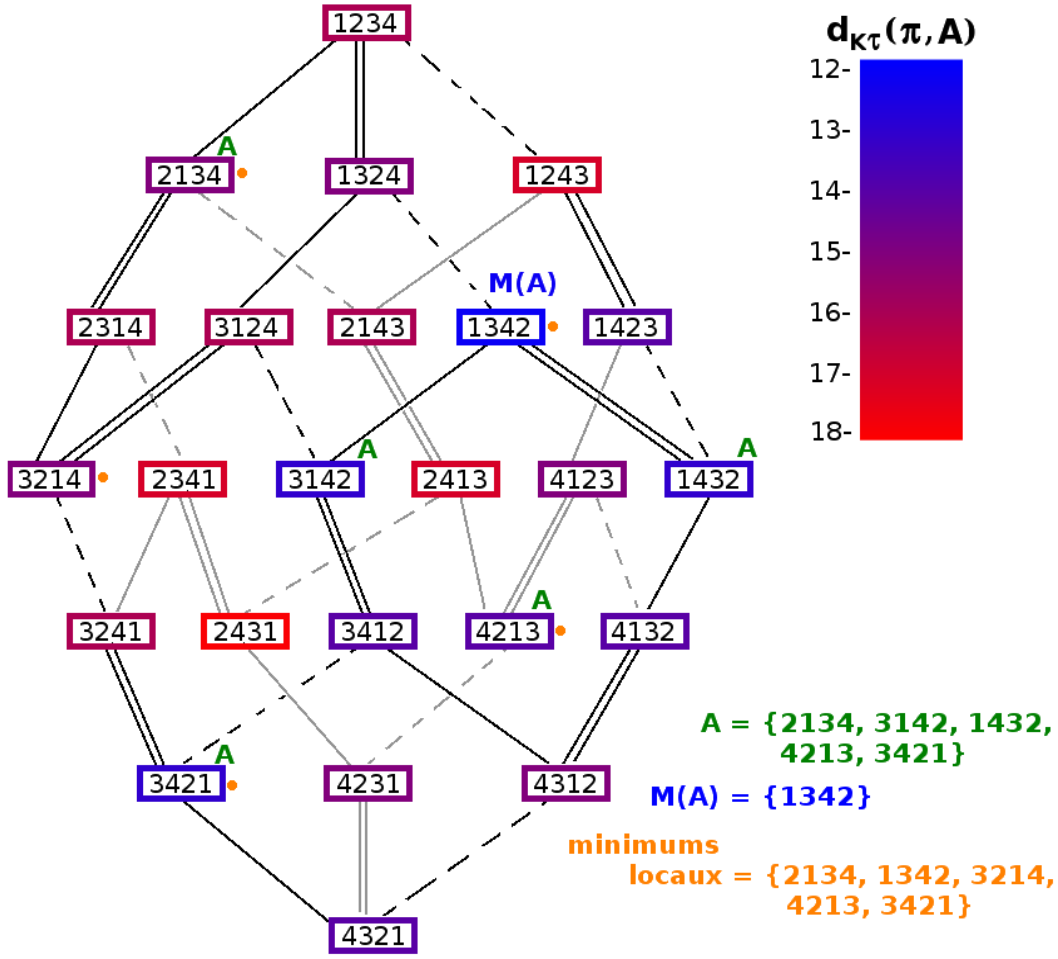


Figure 2.3 – Minimums locaux dans S_4. Ici est représenté S_4 où chaque permutation est coloriée par un gradient de couleur relié à sa distance de l'ensemble de départ (bleu : plus proche, rouge : plus loin). Les permutations de l'ensemble de départ sont notées par un 'A' et la médiane est notée par un 'M(A)'. Les minimums locaux sont notés par des points oranges. Un minimum local est une permutation dont son score (sa distance à A) est inférieur aux scores des permutations voisines (voisines par un swap).

En pratique, on a remarqué une grande différence d'efficacité entre ces mouvements dans les implémentations d'heuristiques et le mouvement circulaire en sort gagnant en terme de temps d'exécution et de rapidité de convergence. De plus, on a remarqué que les permutations de $M(A)$ sont généralement reliées entre-elles par un ou quelques mouvements circulaires, ce qui porte à croire qu'il existe une certaine structure dans l'espace de recherche et que les mouvements circulaires y seraient bien adaptés. Une étude an-

térieure sur le même problème a favorisé ce même mouvement [8]. Une autre étude qui s'attaque à une généralisation du problème (linear ordering problem) avec une approche SA, a choisi le mouvement circulaire et la transposition [12]. Nous avons donc choisi pour toutes ces raisons d'adapter le mouvement circulaire pour le passage d'une permutation courante à une permutation voisine dans notre implémentation. Au final, notre heuristique adaptée au problème de la médiane de permutations peut s'écrire comme le pseudo-code Algortithm 2 de la page 15. Le choix des paramètres est empirique, cela est basé sur les expérimentations (voir prochaine section) et sur le temps aloué pour les calculs sur ordianteur.

2.2.2 Expérimentation et implémentation

La perturbation qui génère la permutation alternative consiste à effectuer un mouvement circulaire (voir Figure 2.2). Il ne faut pas recalculer au complet le score de la permutation alternative, car on peut aisément calculer la différence (Δ) causée par le mouvement circulaire en tenant compte que des ordres relatifs entre l'élément déplacé et les éléments intermédiaires.

Plus il y a de permutations dans l'ensemble de départ A, plus la probabilité d'avoir un espace de recherche constitué de plusieurs bassins d'attraction est grande. C'est pourquoi dans l'implémentation, on répète le processus de SA plusieurs fois avec soit la même solution, soit des solutions de départ différentes. Ce qui va avec l'analogie du recuit dans la vraie vie, où les forgerons des temps anciens ont apprit qu'en répétant le processus de chauffer et refroidir le métal, ils pouvaient fabriquer des épées beaucoup plus dures (les atomes de métal finissaient par se retrouver dans des conformations plus stables).

Dans nos expérimentations, nous avons remarqué que l'efficacité de notre heuristique n'était pas affectée par le choix de notre solution initiale. Cela est sûrement dû à la haute température qu'on met au début de SA dans notre implémentation, ce qui permet pratiquement à la solution courante d'accepter n'importe quels mouvements dans les premiers temps, la rendant assez aléatoire avant la phase de refroidissement où la solution courante va doucement converger vers un certain minimum. Le choix de la température initiale du système reste alors borné par une certainte valeure car il est inutile

Algorithm 2 Simulated Annealing pour la médiane de permutations sous la distance de Kendall-τ

Data: Ensemble A de permutations

Result: Ensemble des meilleures solutions trouvées *BestMA*

begin

$nbElectrons \longleftarrow 5$;

$nbMvts \longleftarrow 1000$;

$\lambda \longleftarrow 0.99$;

$dKTmin \longleftarrow 999999$;

$BestMA \longleftarrow \{\}$;

for $i \leftarrow 1$ ***to*** $nbElectrons$ **do**

$T \longleftarrow 50$;

$\pi \longleftarrow$ permutation aléatoire;

for $i \leftarrow 1$ ***to*** $nbMvts$ **do**

$T \longleftarrow T * \lambda$;

$\pi_{new} \longleftarrow$ mouvement_circulaire_aleatoire(π);

$\Delta \longleftarrow d_{KT}(\pi_{new}, A) - d_{KT}(\pi, A)$;

if $\Delta \leq 0$ **then**

$\pi \longleftarrow permu_{new}$;

if $d_{KT}(\pi, A) = dKTmin$ **then**

$BestMA \longleftarrow BestMA \cup \{\pi\}$

endif

if $d_{KT}(\pi, A) < dKTmin$ **then**

$dKTmin = d_{KT}(\pi, A)$;

$BestMA \longleftarrow \{\}$;

$BestMA \longleftarrow BestMA \cup \{\pi\}$

endif

else

$rand \longleftarrow random[0, 1]$;

if $rand < e^{-\Delta/T}$ **then**

$\pi \longleftarrow \pi_{new}$

endif

endif

endfor

endfor

return $BestMA$;

end

de faire la simulation lorsque toutes les mouvements sont acceptés. Le choix du taux de refroidissement est plus important et il est relié au nombre de mouvements alloués aux électrons.

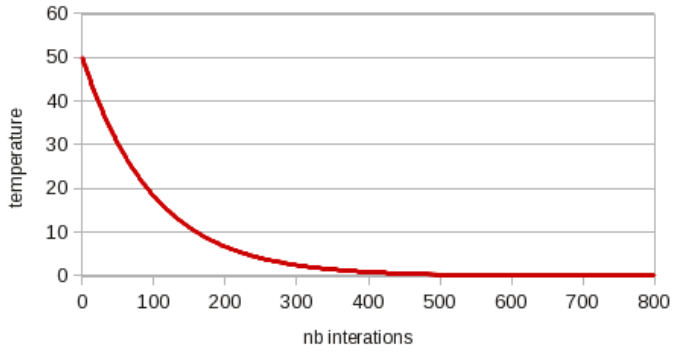


Figure 2.4 – SA : température et refroidissement. Exemple de la Figure 1.1 en application. La température est multipliée par un facteur constant $\lambda = 0.99$ à chaque itération.

Une application de l'heuristique SA présentée dans le pseudo-code Algortithm 2 de la page 15 a été faite sur l'exemple de la Figure 1.1 (modélisé en permutations à la Figure 2.6, $m = 3$ et $n = 8$). On peut voir le graphique de la température à la Figure 2.4 ainsi que le graphique du niveau d'énergie d'un électron (le graphique du score de la permutation courante au fil des mouvements) à la Figure 2.5.

La combinaison des paramètres optimaux pour SA reste inconnue. Seulement une étude a été faite sur des données provenant du domaine de la bio-informatique, dans le contexte généralisé du problème (voir chapitre 5). Les paramètres pour SA ont été choisis de façon arbitraire, mais guidés empiriquement par des expériences. On a observé qu'il est plus efficace de lancer quelques simulations plus courtes qu'une simulation longue, d'où le choix de 'nbElectrons' dans le pseudo-code Algortithm 2. Il est important de souligner que dans le contexte présent on utilise souvent des valeurs assez grandes pour avoir une bonne confiance dans les résultats sur les instances qu'on étudie ($n < 100$). Pour toutes les instances résolvables par algorithme exact, SA trouve en pratique une soltion optimale en quelques secondes ou moins.

On présume, par extrapolation des résultats empiriques du chapitre 5, que SA pourrait trouver une solution optimale rapidement pour des instances $n \sim 100$ dans la majorité

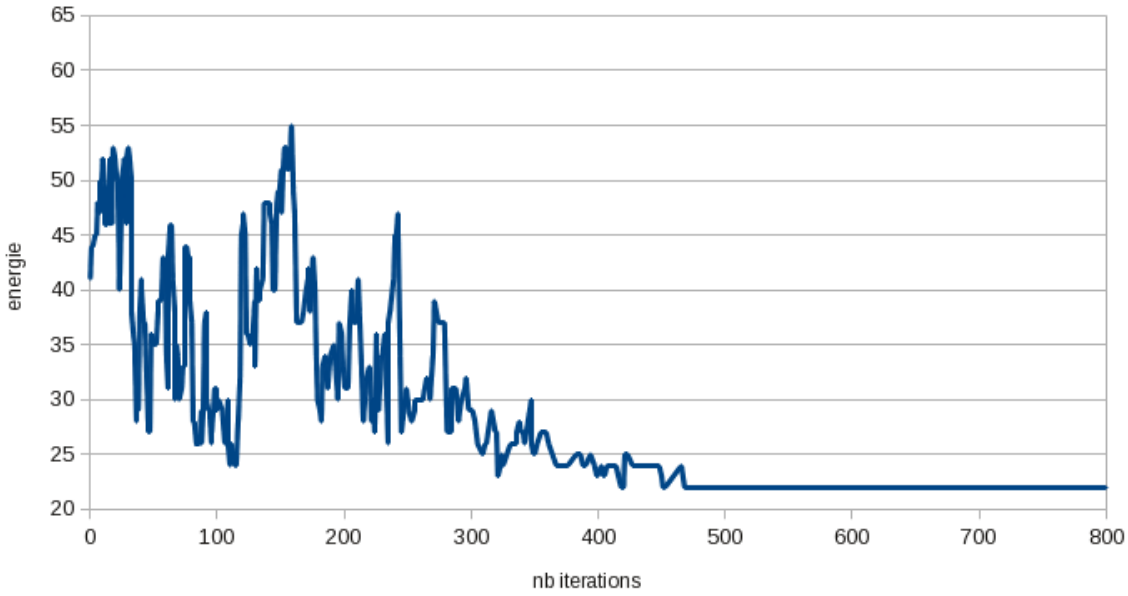


Figure 2.5 – SA : graphique de l'énergie de la solution courante. Exemple en application. On remarque une haute variabilité au début quand la température est haute, suivi d'un refroidissement qui fait converger la solution courante puis un état stable à la fin. Le graphique de la température du système associé à cette simulation se trouve à la Figure 2.4. D'autres simulations sont disponibles dans l'annexe I, Figures I.1,I.2 et I.3.

des cas. Malheureusement, aucun algorithme exacte n'est en mesure de vérifier ces solutions dans ces ordres de grandeurs en un temps raisonnable. À titre indicatif, notre implémentation de SA va prendre 8 secondes pour trouver quelques solutions pseudo-optimales pour une instance de $m = 3$ et $n = 200$ avec 50 électrons et 400000 mouvements accordés par électron.

L'heuristique SA pour le problème de la médiane de permutations a été implémentée en langage Java. C'est présentement la méthode la plus efficace (en terme de temps de calcul) pour résoudre le problème parmis nos algorithmes (SA, Branch and Bound, Label Correcting Algortithm). Sa rapidité est dû à sa simplicité et au fait qu'on peut calculer rapidement la différence de score entre deux permutations séparées par un mouvement circulaire sans recalculer le score global à chaque itération.

2.3 Branch and Bound

2.3.1 Théorie et implémentation

Le principe de l'algorithme du Branch and Bound [27] consiste à faire une recherche en profondeur (arborescence) en coupant les branches dont on sait qu'elles ne contiennent pas une solution optimale. On utilisera un exemple de la Figure 2.6 pour illustrer l'algorithme B&B et l'application de ses contraintes.

```
A = { [ 7, 8, 2, 3, 6, 1, 5, 4],
      [ 3, 5, 1, 7, 8, 6, 2, 4],
      [ 5, 8, 3, 4, 1, 2, 7, 6] }
```

Figure 2.6 – Ensemble de départ *A*, utilisé comme exemple pour cette section. C'est la modélisation à l'aide de permutations du premier exemple illustré en introduction à la Figure 1.1.

Dans notre implémentation, chaque noeud est une solution en construction alors que les feuilles sont des solutions complètes c.-à-d. des permutations. Chaque branche qui descend d'un noeud parent est un choix d'élément à placer à la fin de la permutation en construction.

On ne choisit pas les branches qui vont à l'encontre des contraintes théoriques ni celles qui mènent à des scores non optimaux. Il va de soit que si on coupe plus de branches et plus tôt dans l'arborescence, on explore moins l'espace et on se rend plus rapidement aux solutions optimales. Il est désirable de trouver le plus de contraintes possible pour la coupe. Les conditions d'élagage qui ont été découvertes pour ce problème sont : 1) **gauche/droite**, 2) **semiDist + borneInf** et 3) **contraintes MOT** et sont présentées dans les sections qui suivent.

2.3.2 Contraintes Gauche/Droite

La condition **gauche/droite** vient de [8] et elle consiste au fait que deux éléments adjacents dans une permutation médiane doivent être placés obligatoirement dans leur ordre majoritaire, si l'ordre majoritaire existe bien sûr. Si la permutation en construction se termine par l'élément *b*, on ne peut pas placer un certain élément *a* juste après

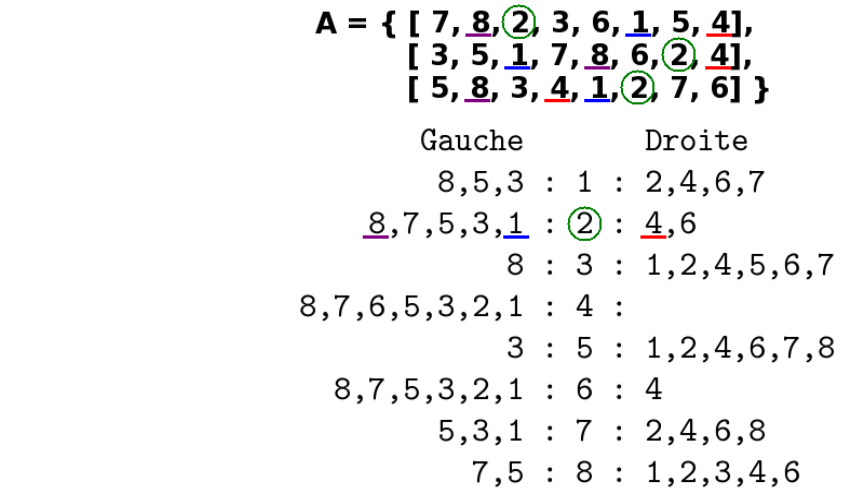


Figure 2.7 – Exemple des contraintes Gauche/Droite pour l'ensemble de la Figure 2.6. Pour chaque paire pour laquelle un ordre majoritaire existe, on définit une contrainte. Dans la figure, on analyse l'élément 2 (encerclé en vert). Avec l'élément 1 (souligné en bleu) cette paire a un ordre majoritaire $1 \prec 2$. Lors de la construction des permutations médianes, si les éléments 1 et 2 sont adjacents, ils seront obligatoirement dans l'ordre $1 \prec 2$ par cette contrainte. En fait, on ne va pas permettre de placer le 1 juste après le 2. Symétriquement, avec l'élément 4 (souligné en rouge) la paire $(2,4)$ dont l'ordre est $2 \prec 4$ ne pourra qu'être dans cet ordre si 2 et 4 sont adjacents. Il peut arriver qu'une paire a un ordre majoritaire total comme le 2 avec le 8 (souligné en mauve), dans ce cas, outre la contrainte Gauche/Droite, l'ordre majoritaire va être obligatoire même si la paire n'est pas adjacente. Il est possible qu'un ordre majoritaire n'existe pas dans une paire (non illustré ici), cela arrive que dans les cas m pair, il n'y a donc pas de contrainte créée pour cette paire.

si l'ordre majoritaire est $a \prec b$. On coupe alors toutes les branches qui proposent des éléments qui ne respecteraient pas l'ordre majoritaire avec le dernier élément placé. La Figure 2.7 illustre la construction de ces contraintes pour l'ensemble A étudié de la Figure 2.6.

2.3.3 Contraintes semiDist + borneInf

La condition **semiDist + borneInf** établit que si le score potentiel d'une permutation en construction est borné par un nombre qui est strictement supérieur au score minimal trouvé jusqu'à présent, alors cette permutation en construction n'est plus intéressante, (Voir Figure 2.8). On coupe alors cette branche pour éviter de s'aventurer dans une arborescence où toutes les permutations finales ne seront pas optimales. Cette borne infé-

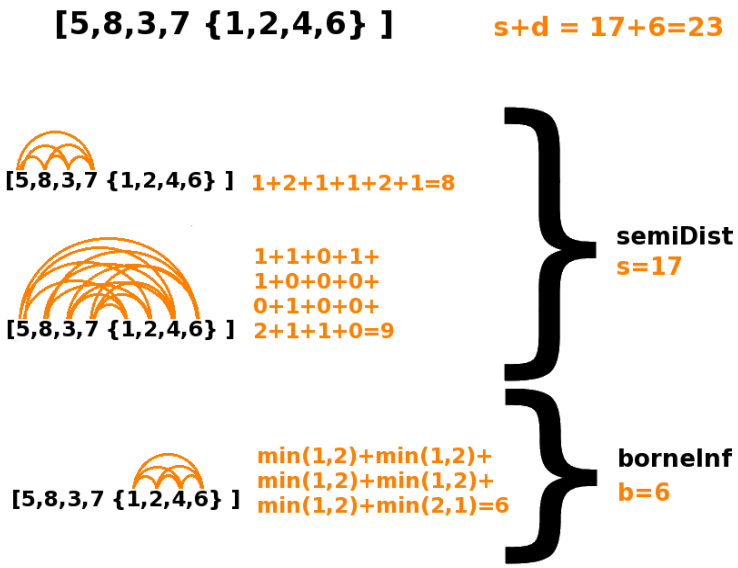


Figure 2.8 – Calcul de la contrainte semiDist + borneInf pour un exemple de permutation en construction. L'ensemble de départ A est celui de la Figure 2.6. On connaît déjà l'ordre des éléments $5,8,3,7$ donc leur apport au score (+8). On sait aussi que ces éléments sont à gauche de ceux qui restent à placer $\{1,2,4,6\}$ (+9). Finalement, l'apport des éléments à placer ne peut pas être plus petit que la somme des coûts pour chaque paire si celles-ci sont toutes placées dans leur ordre majoritaire (+6). On établit alors que n'importe quelle permutation qui commence par "$[5,8,3,7]$" aura un score d'au moins 23. L'exemple vient de la Figure 2.11, il y est encerclé en orange.

rieure sur le score potentiel est la somme de deux variables : *semiDist* (la semi-distance) qui est la part qu'on connaît de la distance de Kendall-τ par les éléments déjà placés puis $borneInf$ (la borne inférieure), une borne théorique sur les éléments qui restent à placer (la somme des coûts minimaux pour chaque paire). Une variante de l'algorithme B&B consiste à non seulement couper les branches dont le score potentiel est borné par un nombre strictemenent supérieur au score de la meilleure solution trouvée jusqu'à présent mais aussi couper les branches cette borne inférieure sur le score potentiel est égale au score de la meilleure solution trouvée jusqu'à présent. On considère ainsi seulement les branches qui peuvent améliorer le score. Cela va accélérer les calculs pour trouver le score optimal (avec une permutation optimale). Par contre, cet algorithme ne va pas nous donner toutes les permutations qui ont un score égal au score de la solution optimale car il aura coupé ces branches. Cette variante nous donne une solution optimale

plus rapidement mais ne retourne pas l'ensemble de toutes les solutions optimales.

2.3.4 Contraintes "Major Order Theorem"

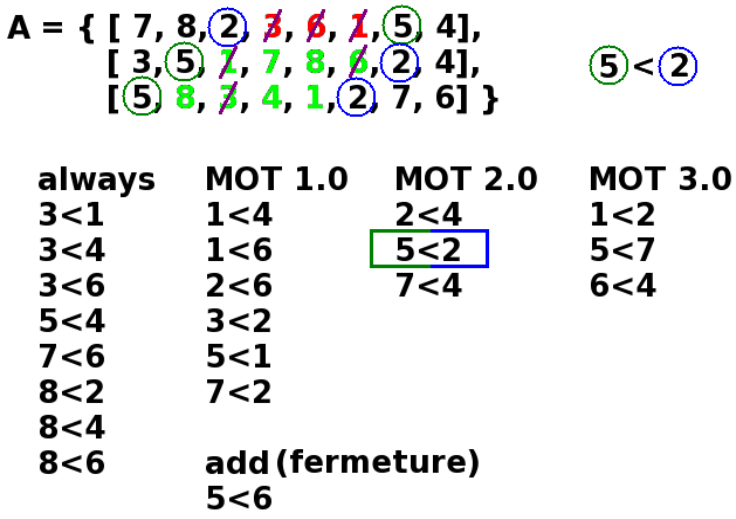


Figure 2.9 – Contraintes Major Order Theorem appliquées sur l'ensemble de la Figure 2.6, 21/28 paires (75%) ont été résolues. Dans cette figure on montre la paire $(5,2)$ qui est ordonnée par le MOT version 2.0 (voir chapitre 3) et toutes les contraintes trouvées par les différentes versions de MOT. La colonne "Always" représente les paires qui sont toujours dans le même ordre dans les permutations de A. La version 1.0 du MOT inclue les contraintes "Always" et chaque version MOT inclue les contraintes des versions précédentes. La fermeture transitive est appliquée sur les contraintes trouvées à chaque étape.

La condition **contrainte MOT** vient du chapitre 3, où l'ordre relatif de certaines paires d'éléments est fixé théoriquement. Si l'ordre $a \prec b$ a été trouvé dans les contraintes MOT, alors on ne peut pas mettre l'élément b à la fin de la permutation en construction si l'élément a ne s'y retrouve pas encore. On coupe alors toutes les branches qui mènent au non-respect des contraintes MOT. Un exemple de calcul des contraintes MOT est montré à la Figure 2.9 à titre indicatif pour la résolution du problème montré dans la figure 2.6. On peut voir la force d'élagage de ces contraintes à la figure 2.11. Il est à noter que la performance du B&B est très sensible au taux de résolution par les contraintes MOT, faisant changer l'ordre de grandeur du temps d'exécution (voir Figure 2.10).

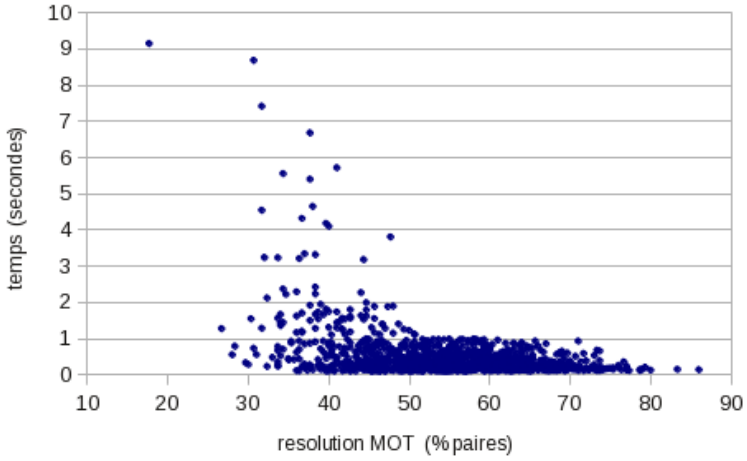


Figure 2.10 – Le temps d'exécution du B&B pour résoudre un problème par rapport au % de résolution par le prétraitement des contraintes par MOT. En général, plus le prétraitement réussi à trouver de contraintes, plus rapide sera la résolution par B&B pour ce problème. Statistiques basées sur 1000 simulations de $m = 3$ et $n = 25$. On peut remarquer qu'à partir de 51% de résolution par MOT, le temps du B&B ne dépasse plus 1.000 secondes, on sait aussi que la moyenne empirique de résolution du MOT pour $m = 3$ et $n = 25$ est de 53.78% (voir chapitre 3). Pour $m = 5, n = 25$ voir Figure I.4 l'annexe I. Avec un faible taux de résolution par MOT, la variance pour le temps du B&B est énorme et on peut obtenir des temps de calcul d'un autre ordre de grandeur comparativement au temps de calcul moyen.

.

2.3.5 Borne supérieure et recherche guidée

On utilise aussi l'heuristique SA (expliquée précédemment) pour obtenir une solution approximative et une distance approximative. En pratique, pour la taille des instances qui sont résolvable par l'algo B&B, SA va nous donner la solution optimale avec un temps négligeable par rapport au temps de calcul de la recherche exacte. Cependant, cette permutation reste une approximation. Le score de cette permutation devient alors le premier score minimal, soit la borne supérieure, enregistré et contribuera à élaguer plus rapidement les branches non prometteuses du B&B. D'une autre part, la permutation elle-même servira à ordonner le choix des éléments à placer lors des embranchements (voir l'exemple de la Figure 2.11). Ainsi, la première permutation atteinte par le B&B sera cette même permutation trouvée par SA. L'ordonnancement des éléments par une heuristique est pertinent, car même si l'ordre n'est pas exactement celui d'une solution optimale, les éléments se retrouvent rapidement dans des places d'intérêts et la recherche

commence directement dans une région hautement pertinente de l'espace de recherche.

2.3.6 Expérimentation et efficacité

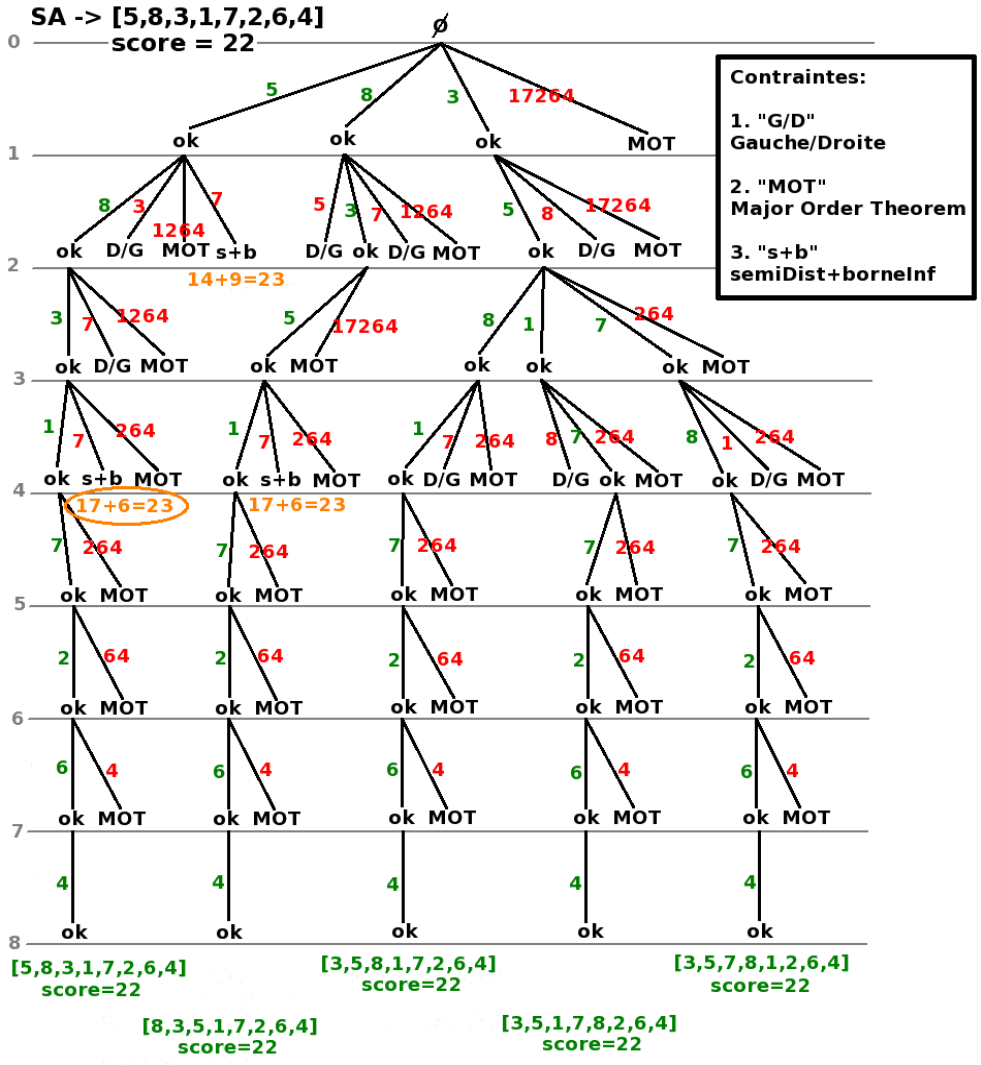


Figure 2.11 – Démonstration du Branch & Bound pour la recherche de l'ensemble médiane de l'ensemble A de la Figure 2.6. La recherche se fait en profondeur et de gauche à droite. Les éléments en vert sont les éléments acceptés alors que ceux en rouge ont été coupés par les contraintes énoncées plus tôt. On trouve l'ensemble des contraintes MOT à la Figure 2.9, l'ensemble des contraintes Gauche/Droite à la Figure 2.7 puis un exemple détaillé de la contrainte semiDist+borneInf, encerclée en orange, à la Figure 2.8. La permutation d'approximation faite par SA se trouve en haut à gauche. On remarque la répétition de la sous-arborescence terminale. L'ensemble $M(A)$ est composé des 5 permutations en vert de score 22.

Plusieurs améliorations sont évidemment encore possibles pour cette approche. Premièrement, resserrer la $borneInf$ de la condition **semiDist + borneInf**. Elle est très simple présentement et pourrait être remplacée par une autre formule légèrement plus élaborée qui tiendrait compte des cycles. On sait [13] [11] que quand il y a un cycle (une suite d'éléments $i_1, i_2, \ldots i_k$ tels que les ordres majoritaires sont $i_j \prec i_{j+1}, j \in \{1 \ldots k-1\}$ et $i_k \prec i_1$) alors la borne $borneInf$ peut être augmentée, car il est impossible que toutes les paires soient simultanément arrangées dans leurs ordres majoritaires respectifs. Deuxièmement, il y a beaucoup de calculs répétés dans l'arborescence du B&B qui pourraient être évités. L'exemple de l'application du B&B de la Figure 2.11 le démontre bien (la sous-arborescence terminale $[..., 2, 6, 4]$ qui se répète). Un modèle déjà proposé par Vincent Antaki (stagiaire d'été) est bien plus efficace et très prometteur, basé sur le Label Correcting Algortithm (article à venir). Il peut aussi être intéressant de garder en mémoire des scores associés au placement de certains éléments et de les utiliser pour mieux estimer la borne minimale d'une permutation en construction.

Sur le plan de l'efficacité, B&B peut être étudié dans trois situations différentes : le "cas $m = 3$", donc seulement trois permutations, le "cas pair" quand $m \geq 4$ et le "cas impair" quand $m \geq 5$. Dans le premier cas, l'efficacité est la plus élevée, suivie du cas impair, puis finalement du cas pair qui donne du fil à retordre aux chercheurs pour des instances de taille similaire. Le cas pair est le seul dont on connaît la complexité théorique. Le cas $m = 3$ est le cas qui nous offre le plus de contraintes MOT. Notez que dans le cas pair, les instances avec $m = 4$ sont généralement les plus difficiles à résoudre. (Voir tableau 2.I)

2.4 Caractéristiques des ensembles de médianes

Afin de mieux comprendre l'efficacité de l'algorithme B&B et de l'heuristique SA ainsi que le choix du mouvement circulaire pour SA, il peut être intéressant de mettre en lumière quelques caractéristiques des ensembles $M(A)$ des permutations médianes.

D'après des expériences avec les algorithmes B&B et SA, on a observé que les permutations médianes sont souvent liées entre elles par un ou quelques mouvements circu-

$m\backslash n$	8	10	12	14	15	20	25	30
3	8	8	8	10	10	18	435	19396
4	10	65	293	36874	114833	-	-	-
5	8	8	8	10	10	23	891	58013
6	9	11	29	2926	1129	-	-	-
10	9	10	11	19	35	4411	-	-
15	8	9	9	10	10	25	788	-
20	9	10	10	12	14	75	-	-
25	9	9	9	11	11	28	658	-

Tableau 2.I – Tableau du temps moyen d'exécution du B&B en millisecondes pour des instances aléatoires de A de m permutations de taille n. Notez la différence entre $m = 3$ et $m = 4$, et la différence générale entre m impair et m pair. Cette différence s'explique en grande partie par le tableau 2.II qui compte le nombre moyen de médianes. Statistiques générées par simulations sur 100 à 1000 instances aléatoires. Pour le nombre exact d'instances calculées pour chaque cas, voir le Tableau I.I en annexe I. (- dénote une donnée non disponible)

laires (voir Figure 2.2). Cela a favorisé le choix de ce mouvement dans l'heuristique SA comme mentionné dans la section 2.2. Un exemple à la Figure 2.12 montre cette propriété. Une autre observation vient de la visualisation de l'ensemble des permutations médianes : on peut remarquer des colonnes d'éléments qui sont conservés (voir Figure 2.12). Ce sont des éléments qui gardent la même position dans chaque permutation médiane ainsi que leur ordre relatif avec les autres éléments. Pouvoir identifier ces éléments stables pourrait être avantageux dans le contexte d'accélérer le calcul des algorithmes.

Une autre caractéristique est le nombre de permutations présentes dans l'ensemble $M(A)$. Des statistiques compilées de simulations nous montrent qu'il y a une différence énorme entre les ensembles $M(A)$ des instances de A avec m pair et les instances de A avec m impair, le cas pair donnant généralement beaucoup plus de permutations médianes (voir tableau 2.II). Ces données expliquent la différence de temps d'exécution pour le B&B entre les cas pairs et les cas impairs (voir tableau 2.I).

Une dernière caractéristique est la parité du nombre de médianes. Encore une fois des statistiques nous montrent que le cas pair a généralement plus de chance d'avoir un ensemble $M(A)$ de taille paire alors que pour le cas impair, il y a plus de chance d'avoir un ensemble $M(A)$ de taille impaire, (voir tableau 2.III). Pour des ensembles de

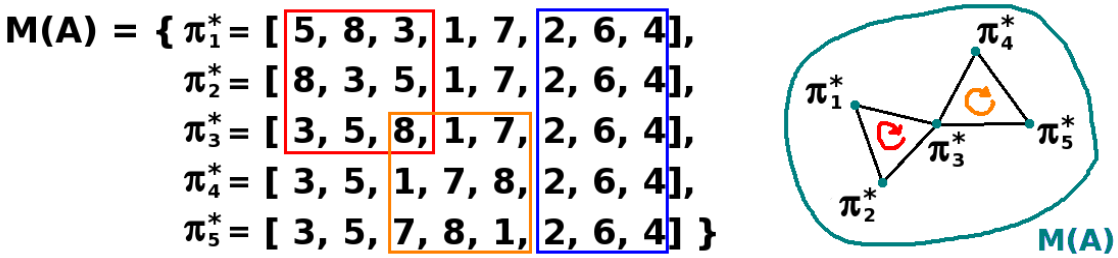


Figure 2.12 – Exemple de $M(A)$, l'ensemble de permutations médianes pour l'exemple étudié de la Figure 2.6. Il y a deux faits importants à observer. Premièrement, certains éléments conservent toujours le même ordre alors que d'autres ne le conservent pas. Dans l'exemple illustré, les éléments $2, 6, 4$, encadrés en bleu, gardent les mêmes positions dans toutes les permutations médianes alors que les autres éléments $(5, 8, 3, 1, 7)$ changent de position. Deuxièmement, l'ensemble $M(A)$ adopte une structure où il est possible de naviguer entre les permutations du groupe π_1^*, π_2^* et π_3^* par un seul mouvement circulaire (illustré en rouge) et de naviguer aussi entre les permutations du groupe π_3^*, π_4^* et π_5^* par un seul mouvement circulaire (illustré en orange) alors que la permutation π_3^* relie ces deux groupes. L'ensemble $M(A)$ de cet exemple est connexe sous l'opération du mouvement circulaire.

permutations de taille n, n petit, la probabilité d'avoir un nombre impair de permutations dans $M(A)$ est plus élevée alors que pour n plus grand, ce phénomène s'inverse.

$m\backslash n$	8	10	12	14	15	20	25	30
3	2.1	3.0	3.7	4.8	5.6	12.2	23.1	61.4
4	60.6	331.4	1321.4	7551.4	14253.8	-	-	-
5	2.2	2.9	3.6	5.2	6.2	12.9	29.1	49.2
6	31.3	90.6	345.1	1506.2	1614.9	-	-	-
10	13.0	36.8	88.8	201.9	315.6	2947.9	-	-
15	1.7	2.2	2.8	3.5	3.8	6.3	12.3	-
20	6.3	11.4	22.2	39.8	55.5	256.7	-	-
25	1.6	1.9	2.3	2.6	2.9	4.6	7.6	-

Tableau 2.II – Tableau du nombre moyen de permutations médianes dans $M(A)$ pour des instances aléatoires de A de m permutations de taille n. Statistiques générées par simulations sur 100 à 1000 instances aléatoires.Pour le nombre exact d'instances calculées pour chaque cas, voir le Tableau I.I en annexe I. (- dénote une donnée non disponible)

$m\backslash n$	8	10	12	14	15	20	25	30
3	0.92	0.83	0.78	0.75	0.72	0.61	0.55	0.56
4	0.29	0.24	0.22	0.18	0.18	-	-	-
5	0.80	0.73	0.68	0.62	0.56	0.52	0.40	0.30
6	0.33	0.28	0.25	0.25	0.21	-	-	-
10	0.40	0.33	0.30	0.25	0.24	0.17	-	-
15	0.74	0.69	0.59	0.55	0.54	0.42	0.37	-
20	0.41	0.36	0.29	0.31	0.28	0.20	-	-
25	0.74	0.67	0.62	0.58	0.53	0.44	0.35	-

Tableau 2.III – Tableau de la probabilité d'avoir un nombre impair de permutations médianes dans $M(A)$. Chaque case indique la probabilité que la taille de $M(A)$ soit impaire, pour des instances aléatoires de A de m permutations de taille n. Statistiques générées par simulations sur 100 à 1000 instances aléatoires. Pour le nombre exact d'instances calculées pour chaque cas, voir le Tableau I.I en annexe I. (- dénote une donnée non disponible)

CHAPITRE 3

MAJOR ORDER THEOREM

3.1 Résumé

L'article propose un théorème en une cascade de versions qui trouvent des contraintes de plus en plus fortes pour fixer l'ordre entre certaines paires d'éléments, résolvant ainsi une partie du problème et réduisant l'espace de recherche pour la résolution complète du problème.

Il y a peu de support théorique pour la portée du théorème, toutefois des simulations statistiques démontrent clairement l'intérêt de la méthode en plus de la comparer à deux autres approches trouvées dans la littérature.

L'article "Medians of permutations : building constraints" a été accepté à la conférence Conference on Algorithms and Discrete Applied Mathematics, February 2016, Thiruvanthapuram, Kerala, Inde (CALDAM2016) et se retrouvera dans les proceedings de celle-ci [29].

3.2 Partage du travail

Dans cet article, j'ai développé et prouvé le théorème "Major Order Theorem" 1-2-3 (esquisses de preuves pour MOT 2-3). J'ai implémenté le théorème en code Java et j'ai construit des statistiques par simulation à partir d'un grand nombre d'instances du problème. J'ai aussi aussi implémenté ce théorème dans un algo exacte B&B. Sylvie Hamel et moi avons rédigé l'article.

Medians of permutations : building constraints *

Robin Milosz and Sylvie Hamel

DIRO - Université de Montréal ,
C. P. 6128 Succ. Centre-Ville, Montréal, Qc, Canada, H3C 3J7,

Abstract. Given a set $\mathcal{A} \subseteq \mathcal{S}_n$ of m permutations of $[n]$ and a distance function d, the **median** problem consists of finding a permutation π^* that is the "closest" of the m given permutations. Here, we study the problem under the Kendall-τ distance which counts the number of pairwise disagreements between permutations. This problem has been proved to be NP-hard when $m \geq 4$, m even. In this article, we investigate new theoretical properties of $\mathcal{A}$ that will solve the relative order between pairs of elements in median permutations of $\mathcal{A}$, thus drastically reducing the search space of the problem.

1 Introduction

The problem of finding medians of a set of m permutations of $\{1, 2, \ldots, n\}$ under the Kendall-τ distance [10,15] is often cited in the literature as the Kemeny Score Problem [9]. In this problem m voters have to order a list of n candidates according to their preferences. The problem then consists of finding a *Kemeny consensus*: an order of the candidates that agrees the most with the order of the m voters, *i.e.*, that minimizes the sum of the disagreements.

This problem is polynomial-time solvable for $m = 2$, has been proved to be NP-complete when $m \geq 4$, m even (first proved in [6], then corrected in [4]), but its complexity remains unknown for $m \geq 3$, m odd. In the last 10 years, different approximation algorithms have been derived. First, a randomized algorithm with approximation factor 11/7 [1] and then a deterministic one with approximation factor 8/5 [16] were designed. In 2007, a PTAS result was obtained [11] and some years later, different fixed-parameter algorithms have been described [3,8,13,14].

Solving the median problem may also be seen as solving the order of all the $\frac{n(n-1)}{2}$ possible pairs of elements of a median. Other theoretical approaches working in that direction and aiming at reducing the search space for this problem have also been developed. In [5], a theorem targeting pairwise ordering was proposed along with some constraints about adjacent elements of a median. A data reduction was proposed in [2] where a "non-dirty" element can be ordered with respect to all other elements in a median, splitting the ordering problem into two smaller ones.

* supported by NSERC through an Individual Discovery Grant (Hamel) and by FRQNT through a Master's scholarship (Milosz)

In this present work, we are interested in the theoretical perspective of the problem, investigating properties of an instance that can partly resolve and accelerate computation. We will derive necessary conditions for setting the order of appearance of pairs of elements in a median, building constraints that will drastically reduce the search space for this median.

This article is organized as follows: after introducing the basic definitions and notations in Section 2, we resume, in Section 3, some related previous works aiming at reducing the search space for the medians. Section 4 presents our approach, while Section 5 presents its results on uniformly generated random sets of permutations. Section 5 also compares our approach with the ones resumed in Section 3. Finally, we conclude and give some future directions for this work in Section 6.

2 Median of permutation: definitions and notations

A **permutation** π is a bijection of $[n] = \{1, 2 \ldots, n\}$ onto itself. The set of all permutations of $[n]$ is denoted $\mathcal{S}_n$. As usual we denote a permutation π of $[n]$ as $\pi = \pi_1 \pi_2 \ldots \pi_n$. Let $\mathcal{A} \subset \mathcal{S}_n$ be a set of permutations of $[n]$, we will denote its cardinality by $\#\mathcal{A}$.
The **Kendall-τ distance**, denoted d_{KT}, counts the number of pairwise disagreements between two permutations and can be defined formally as follows: for permutations π and σ of $[n]$, we have that

$$d_{KT}(\pi, \sigma) = \#\{(i,j) | i < j \text{ and } [(\pi[i] < \pi[j] \text{ and } \sigma[i] > \sigma[j]) \text{ or } (\pi[i] > \pi[j] \text{ and } \sigma[i] < \sigma[j])]\},$$

where $\pi[i]$ denotes the position of integer i in permutation π.
Given any set of permutations $\mathcal{A} \subseteq \mathcal{S}_n$ and a permutation π, we have

$$d_{KT}(\pi, \mathcal{A}) = \sum_{\sigma \in \mathcal{A}} d_{KT}(\pi, \sigma).$$

The **problem of finding a median of $\mathcal{A}$ under the Kendall-τ distance** can be stated formally as follows: Given $\mathcal{A} \subseteq \mathcal{S}_n$, we want to find a permutation π^* of $\mathcal{S}_n$ such that $d_{KT}(\pi^*, \mathcal{A}) \leq d_{KT}(\pi, \mathcal{A}),\ \forall\ \pi \in \mathcal{S}_n$. Note that a set $\mathcal{A}$ can have more than one median.

3 Previous approaches

When dealing with permutations, searching through the whole set of permutations $[n]$ quickly becomes impossible since there are $n!$ such permutations. To be able to find exact medians for sets of "big" permutations, we need to reduce the search space so that the computation will take place in a reasonable time.

Here, given a set of permutations $\mathcal{A} \subseteq \mathcal{S}_n$, we present two previous approaches that reduce the search space by discarding non relevant permutations.

3.1 Data reduction with non-dirty candidates

In [2] a **non-dirty pair** of candidates according to a certain threshold $s \in [0, 1]$ is a pair of elements (a, b), $a, b \in [n]$, which respect the following property: either a is favored to b [1] in a ratio of s or more in the starting set of permutations $\mathcal{A}$, or b is favored to a in a ratio of s or more in the starting set $\mathcal{A}$. A **non-dirty candidate** is a element which forms a non-dirty pair with every other element of $[n]$ according to the threshold s.

It has been proven in [2] that with $s = 3/4$, elements of a median permutation will be necessarily ordered relatively to a non-dirty candidate in the majority order. In other words, a non-dirty candidate will separate the median permutation such as all the elements that are favored to it will be to its left and all the elements that are not will be to its right. In what follows, we will refer to this result as the *3/4 majority rule*.

Having such a non-dirty candidate allows to cut the instance of the problem in two sub-instances, on which the approach can be re-applied. It also allows a parallelization of the problem because the two sub-instances are independent. Betzler's *et al.* approach is strong on sets of permutations derived from real data, since, in those cases, the permutations are often really close to each other, greatly increasing the probability of finding non-dirty candidates.

The drawback is that non-dirty candidates are rare on uniformly generated random sets of permutations, like shown in Table 2 and discussed in Section 5.

3.2 Always theorem

In [5] a pairwise ordering theorem was described for a pair of elements (a, b) such that if a is favored to b in all permutations of $\mathcal{A}$ then a will be necessarily favored to b in any median permutation. This theorem allows to set the relative order in pairs of elements that respect the property of being always at the right or always at the left of the other element. We will refer to this theorem as the *always theorem*.

The drawback is that the efficiency of this theorem for a set of uniformly distributed random permutations is greatly affected by the number of permutations m in our given set $\mathcal{A}$. The probability that a random pair satisfies the *always theorem* is $p = \frac{1}{2^{m-1}}$ because it has to keep the same order in each of the m permutations and there are two possible orders $a < b$ and $b < a$. For three permutations, $p = 25\%$, for four permutations $p = 12.5\%$ and so on (See Table 1).

4 Our approach

We were inspired by the two approaches resumed in Section 3, with the aim of building ordering contraints for pairs of elements in a median. We ask ourselves if there was a data reduction which was less restrictive than the *3/4 majority rule* but more englobing than the *always theorem*.

[1] element a is place to the left of element b

4.1 Existence of a majority bound?

Given that the *always theorem* guarantees the order of a pair of elements in a median, if this pair is always in that order in all permutations of $\mathcal{A}$, our first idea was to find a %-majority bound, where having an element favored to another one in at least $p\%$ of the permutations of $\mathcal{A}$ would guarantee the order of this pair of elements in a median of $\mathcal{A}$. Unfortunately, no such bound exists.

Proposition 1 *For any given bound s, $0.5 < s < 1$, it is always possible to construct a set $\mathcal{A}$ in which the proportion $p\%$ of permutations favoring element i to element j (the major order) will be at least s, $s \leq p\% < 1$, but for which the order of the pair (i, j) in any median of $\mathcal{A}$ will contradict this majority order.*

Proof See Appendix. 7

4.2 Major Order theorem

Since nothing can be derived from a majority ordering, another idea comes from the observation that two elements that are close enough in all permutations of $\mathcal{A}$ will have the tendency to be placed in their major order, in any median of $\mathcal{A}$, because there is less interference caused by other elements.

The total absence of interference between two elements is found when they are adjacent in all permutations. In that case, we can consider them as one heavier element and their relative order in a median permutation will clearly be the majority order.

So, what happens if we limit the interference between two elements? Can we then have an extension of the *always theorem*? The answer is yes and will be given by our major order theorem below. But first, we need some definitions and notations.

Let $\mathcal{A} \subseteq \mathcal{S}_n$ be a set of m permutations of $[n]$. Let us build, for each pair of elements (i, j), $1 \leq i < j \leq n$, two multisets $E_{ij}(\mathcal{A})$ and $E_{ji}(\mathcal{A})$.

Multiset $E_{ij}(\mathcal{A})$ (resp. $E_{ji}(\mathcal{A})$) will contain all the elements present between elements i and j in all permutations of $\mathcal{A}$, where element i is positioned before (resp. after) element j, denoted $i < j$ (resp. $j < i$). Mathematically, we have

$$E_{ij}(\mathcal{A}) = \bigcup_{\pi \in \mathcal{A}, \pi[i] < \pi[j]} \{k \mid \pi[i] < \pi[k] < \pi[j]\},$$

and

$$E_{ji}(\mathcal{A}) = \bigcup_{\pi \in \mathcal{A}, \pi[j] < \pi[i]} \{k \mid \pi[j] < \pi[k] < \pi[i]\}.$$

When there are no ambiguities, we will simply denote $E_{ij}(\mathcal{A})$ by E_{ij}.

Example 1 *For $\mathcal{A} = \{\underline{1}34\underline{2}5, 4\underline{1}3\underline{2}5, 4\underline{2}35\underline{1}\}$, we have $E_{12} = \{3, 3, 4\}$, which are the elements present in the first two permutations of $\mathcal{A}$, where element 1 is positioned before element 2. $E_{21} = \{3, 5\}$, which are the elements between 2 and 1 in the third permutation.*

To keep track of the number of permutations in $\mathcal{A}$, that have a certain order between two elements, let us introduce the left/right distance matrices L and R.

Definition 1 *Let $L(\mathcal{A})$, or simply L, when there are no ambiguities, be the* **left distance matrix** *of $\mathcal{A}$, where $L_{ij}(\mathcal{A})$ denotes the number of permutations of $\mathcal{A}$ having element i to the left of element j. Symmetrically, let $R(\mathcal{A})$, or simply R, be the* **right distance matrix** *of $\mathcal{A}$, where $R_{ij}(\mathcal{A})$ denotes the number of permutations of $\mathcal{A}$ having element i to the right of element j. Obviously, $L_{ij} + R_{ij} = m$ and $L_{ij} = R_{ji}$.*

Notations: Let $\overrightarrow{L_x}$ be the vector of the left distance matrix L, associated to the element x. Note that $\overrightarrow{L_x}[y] = L_{xy}$. The L_1-norm of the vector $||\overrightarrow{L_x}||_1$, is the sum of the absolute values of all the elements of the vector, so here it represents the sum of all the times where x is to the left of any element in all permutations of $\mathcal{A}$. For S, a set of elements of $[n]$, we define L_{xS} as $L_{xS} = \sum_{s \in S} L_{xs}$, i.e. the total number of times where x is to the left of an element of S in all permutations of $\mathcal{A}$. $\overrightarrow{R_x}$, $||\overrightarrow{R_x}||_1$ and $R_{xS} = \sum_{s \in S} R_{xs}$ are defined symmetrically. Note that $d_{KT}(\pi, \mathcal{A}) = \sum_{i,j|\pi[i]<\pi[j]} L_{ij}$.

Example 2 *In set $\mathcal{A}$ of Example 1, $L_{12} = 2$ and $R_{12} = 1$ since 1 is to the left of (or favored to) 2 in the two first permutations of $\mathcal{A}$ and to the right of 2 only in the last permutation of $\mathcal{A}$. Here, $\overrightarrow{L_1} = [0, 2, 2, 1, 2]$ and $||\overrightarrow{L_1}||_1 = 7$. Let $S = \{2, 4\}$, then $L_{1S} = L_{12} + L_{14} = 3$.*

We are now almost ready to state our major order theorem but first let us formally define the major order between elements.

Definition 2 *We say that the* **major order** *between elements i and j is $i < j$ (resp. $j < i$) if $L_{ij} > R_{ij}$ (resp. $R_{ij} > L_{ij}$), the* **minor order** *is then $j < i$ (resp. $i < j$). We will use d_{ij} to denote the difference between the major and minor order of two elements i and j, $d_{ij} = |L_{ij} - R_{ij}| = |R_{ij} - L_{ij}| = |L_{ji} - R_{ji}| = d_{ji}$.*

Theorem 1 (Major Order Theorem 1.0) *Let $\mathcal{A} \subseteq \mathcal{S}_n$ be a set of permutations of $[n]$. For a pair of elements (i, j), $1 \leq i < j \leq n$, if $i < j$ (resp. $j < i$) is their major order and $d_{ij} > \#E_{ji}$ (resp. $d_{ij} > \#E_{ij}$) then this major order will be conserved in all medians π^* of $\mathcal{A}$.*

Proof of Theorem 1: Suppose, w.l.o.g., that for a pair of elements (i, j) the major order is $i < j$ and that the conditions of Theorem 1 are fulfilled i.e. we have $d_{ij} > \#E_{ji}$. By contradiction, suppose that we have a median permutation π^* for $\mathcal{A}$ in which i and j are in their minor order $j < i$. Let $\pi^* = B\, j\, K\, i\, A$ be such a median, where B, K and A are the sets of elements found before (B), in between (kernel - K) and after (A) elements i and j.

The contribution of element i to the Kendall-τ distance $d_{KT}(\pi^*, \mathcal{A})$ is

$$\underbrace{L_{iB}}_{\substack{\#\text{ of times } i \text{ is left to } b \in B, \\ \text{in all } \sigma \in \mathcal{A}}} + \underbrace{L_{iK}}_{\substack{\#\text{ of times } i \text{ is left to } k \in K, \\ \text{in all } \sigma \in \mathcal{A}}} + \underbrace{R_{iA}}_{\substack{\#\text{ of times } i \text{ is right to } a \in A, \\ \text{in all } \sigma \in \mathcal{A}}} + L_{ij}.$$

Similarly, the contribution of element j to $d_{KT}(\pi^*, \mathcal{A})$ is $L_{jB} + R_{jK} + R_{jA} + L_{ij}$. We will show that for either $\sigma^* = B\,i\,j\,K\,A$ or $\sigma^* = B\,K\,i\,j\,A$, we have $d_{KT}(\sigma^*, \mathcal{A}) < d_{KT}(\pi^*, \mathcal{A})$ contradicting our choice of median and, at the same time, our choice of ordering for the pair (i, j).

We will investigate interactions between i, j and elements of the set K, since the elements of the sets A and B stay in the same relative order with i and j in π^* and either choice of σ^*. (Elements in A (resp. B) will always be after (resp. before) elements i and j).

The cost of the interaction of i with K and of j with K is respectively L_{iK} and R_{jK} in the supposed median π^*. Thus, there are two possible cases: either $L_{iK} \leq R_{jK}$ or $L_{iK} > R_{jK}$.

Case 1: $\mathbf{L_{iK} \leq R_{jK}}$

For this case, let $\sigma^* = B\,K\,i\,j\,A$, i.e. we moved element j at the immediate right of element i in our supposed median $\pi^* = B\,j\,K\,i\,A$. In this case, we have

$$\begin{aligned}
d_{KT}(\sigma^*, \mathcal{A}) - d_{KT}(\pi^*, \mathcal{A}) &\overset{(\diamond_1)}{=} (L_{jB} + L_{jK} + R_{jA} + R_{ij}) - (L_{jB} + R_{jK} + R_{jA} + L_{ij}) \\
&= (L_{jK} + R_{ij}) - (R_{jK} + L_{ij}) \\
&= L_{jK} - R_{jK} + -(L_{ij} - R_{ij}) \\
&= L_{jK} - R_{jK} - d_{ij} \\
&\overset{(*_1)}{\leq} L_{iK} + \#E_{ji} - R_{jK} - d_{ij} \\
&\overset{Case\,1}{\leq} R_{jK} + \#E_{ji} - R_{jK} - d_{ij} \\
&= \#E_{ji} - d_{ij} \\
&< 0.
\end{aligned}$$

The last inequality comes from the initial condition of the theorem. Equality $(\diamond_1)$ is obtained by taking into account only the contribution of the element that changes position between π^* and σ^*, i.e. element j, since the contribution of the other elements will cancel each other out. As for inequality $(*_1)$, it comes from the fact that $L_{jK} \leq L_{iK} + \#E_{ji}$ since for an element $k \in K$, we will add one to L_{jk} iff j is to the left of k in a permutation of $\mathcal{A}$. In this same permutation, either i is also to the left of k (captured by adding one to L_{ik}) or to the right, in which case, k is an element of E_{ji}.

Thus, moving j after i gives us a permutation σ^* which is closer to the set $\mathcal{A}$ contradicting our choice of π^* as a median of $\mathcal{A}$.

Case 2: $\mathbf{L_{iK} > R_{jK}}$

For this case, let $\sigma^* = B\,K\,i\,j\,A$, i.e. we moved element i at the immediate left of element j in our supposed median $\pi^* = B\,j\,K\,i\,A$. In this case, we have

$$
\begin{aligned}
d_{KT}(\sigma^*,\mathcal{A}) - d_{KT}(\pi^*,\mathcal{A}) &\overset{(\diamond_2)}{=} (L_{iB} + R_{iK} + R_{iA} + R_{ij}) - (L_{iB} + L_{iK} + R_{iA} + L_{ij}) \\
&= (R_{iK} + R_{ij}) - (L_{iK} + L_{ij}) \\
&= R_{iK} - L_{iK} + -(L_{ij} - R_{ij}) \\
&= R_{iK} - L_{iK} - d_{ij} \\
&\overset{(*_2)}{\leq} R_{jK} + \#E_{ji} - L_{iK} - d_{ij} \\
&\overset{Case\,2}{<} L_{iK} + \#E_{ji} - L_{iK} - d_{ij} \\
&= \#E_{ji} - d_{ij} \\
&< 0.
\end{aligned}
$$

Again, the last inequality comes from the initial condition of the theorem. Equality $(\diamond_2)$ is obtained by taking into account only the contribution of the element that changes position between π^* and σ^*, i.e. element i. As for inequality $(*_2)$, it comes, by symmetry, from $(*_1)$, i.e. $R_{iK} \leq R_{jK} + \#E_{ji}$.

In each of the two cases, we were able to find a permutation σ^*, such that $d_{KT}(\sigma^*,\mathcal{A}) < d_{KT}(\pi^*,\mathcal{A})$, contradicting our choice of median π^* and our choice of ordering for the pair of elements (i,j). Consequently, i and j can only be placed in their major order $i < j$ in a median permutation if the conditions are fulfilled. ■

4.3 Refined versions of the major order theorem

We tested our major order theorem 1.0 on randomly generated sets of permutations (see Table 1) and saw that its efficiency, in terms of the number of pairs of elements ordered, was not that much better than the *always theorem*, as n grows. To be able to solve the ordering of a bigger number of pairs of elements, we needed to find a way to reduce the size of our multisets E_{ij} and E_{ji}.

Refined version 1. We observed that the presence of an element k in both multisets E_{ij} and E_{ji} cancels its impact on the ordering of the pair of elements (i,j). This leads to a 2.0 version of our major order theorem presented below.

Let $E'_{ij}(\mathcal{A})$ and $E'_{ji}(\mathcal{A})$ be those new multisets defined by $E'_{ij}(\mathcal{A}) = E_{ij}(\mathcal{A}) \setminus E_{ji}(\mathcal{A})$ and $E'_{ji}(\mathcal{A}) = E_{ji}(\mathcal{A}) \setminus E_{ij}(\mathcal{A})$.

Example 3 *In set $\mathcal{A}$ of Example 1, $E'_{12} = E_{12}(\mathcal{A}) \setminus E_{21}(\mathcal{A}) = \{3,3,4\} \setminus \{3,5\} = \{3,4\}$ and $E'_{21} = E_{21}(\mathcal{A}) \setminus E_{12}(\mathcal{A}) = \{3,5\} \setminus \{3,3,4\} = \{5\}$.*

Theorem 2 (Major Order Theorem 2.0) *Let $\mathcal{A} \subseteq \mathcal{S}_n$ be a set of permutations of $[n]$. For a pair of elements (i,j), $1 \leq i < j \leq n$, if $i < j$ (resp. $j < i$) is their major order and $d_{ij} > \#E'_{ji}$ (resp. $d_{ij} > \#E'_{ij}$) then this major order will be conserved in all medians π^* of $\mathcal{A}$.*

Sketch of proof. In an informal way, let us pretend w.l.o.g. that the majority order between i and j is $i < j$. We observed that every element k found between i and j in a permutation where i and j are in the minor order $j < i$ (i.e. $k \in E_{ji}$), will increase L_{jk} by one but will not increase L_{ik}. Consequently, it will increase $||\overrightarrow{L_j}||_1$ by one but not increase $||\overrightarrow{L_i}||_1$ and it will increase L_{jS} by one but not increase L_{iS} for those sets S that contain k.

If this element k is also found between i and j in another permutation where i and j are in the major order $i < j$ (i.e. $k \in E_{ij}$), it will increase L_{ik} by one but will not increase L_{jk}. Consequently in the same way, it will increase $||\overrightarrow{L_i}||_1$ by one but not increase $||\overrightarrow{L_j}||_1$ and it will increase L_{iS} by one but not increase L_{jS} for those sets S that contain k. So, if this happens, it will cancel the contribution of k in E_{ji}.

Because only elements of E_{ji} have an impact on the upper bound when $i < j$ is the major order, we are interested to minimize the size of E_{ji}. We can cancel one copy of k in E_{ji} if a copy of k is found in E_{ij} because each copy cancels the effect of the other side's copy in the difference between L_{ik} and L_{jk}. We take out one copy of k in both multisets and repeat the process with every common element until the intersection of those two collections becomes empty. ■

This new version 2.0 of our major order theorem was also tested on randomly generated sets of permutations (see Table 1) and its efficiency, discussed in more details in Section 5, is much better then our 1.0 version even as n grows.

Refined version 2. Given a set of permutations $\mathcal{A}$, Theorem 2 gives us a set of ordering constraints for some pairs of elements in a median of $\mathcal{A}$. We can use this set to extend the reach of the theorem by applying a second filter over the multisets E'_{ij}, E'_{ji}, in the following way: While investigating a pair of elements (i, j), if we previously found a pair of constrains related to an element k, ($k < i$ and $k < j$) or ($i < k$ and $j < k$), then this element k cannot be found in between i and j in a median permutation. Regarding the proof of Theorem 1, with $\pi* = BjKiA$, it cannot be in K, thus its contribution to L_{iK}, L_{jK}, R_{iK} and R_{jK} is null and so it can be removed from E'_{ij} and E'_{ji}. In this way, we can trim the multisets E'_{ij} and E'_{ji} by taking out all copies of elements that have been proved, by our ordering contraints, to be to the right, or to the left, of both i and j. Let ${E''_{ij}}^1$ and ${E''_{ji}}^1$ be these new trimmed multisets, obtained after this first iteration. We continue applying the same theorem but upgrading the $d_{ij} > \#E'_{ji}$ condition to $d_{ij} > \#{E''_{ji}}^1$. We iterate this process, obtaining at iteration t, new trimmed sets ${E''_{ij}}^t$ and ${E''_{ji}}^t$. We stop the process when an iteration does not find any new constraint. This gives us the following refined version of Theorem 2:

Theorem 3 (Major Order Theorem 3.0) *Let $\mathcal{A} \subseteq \mathcal{S}_n$ be a set of permutations of $[n]$. For a pair of elements (i, j), $1 \leq i < j \leq n$, if $i < j$ (resp. $j < i$) is their major order and $d_{ij} > \#{E''_{ji}}^t$, $t \in \mathbb{N}$ (resp. $d_{ij} > \#{E''_{ij}}^t$) then this major order will be conserved in all medians π^* of $\mathcal{A}$.* ■

5 Efficiency of our approach

In this section, we will present the efficiency of our approach on randomly generated data and also compare it to the previous approaches briefly described in Section 3. In what follows, n will represent the number of elements in our permutations and m the size of the set of permutations considered.

We will base our efficiency statistics on the proportion of solved ordering of pairs of elements. (For permutations of $[n]$, there are $\frac{n(n-1)}{2}$ pairs to order.)

Note that for efficiency and theoretical concerns, after applying any theorem on a set of permutations $\mathcal{A}$, we apply the transitive closure on the sets of constraints found, since if we know that an element i is to the left of j and element j is to the left of k, than we also know that element i will be left of k. (The *always theorem* is always transitive closed but it is not the case for our major order theorems.)

We evaluated our approach on uniformly generated random permutation sets. For each couple $m,\ n$, statistics where calculated over 2000 (big n) to 100000 (smaller n) instances. Fisher-Yates shuffle also known as Knuth shuffle [7,12] was used to create random permutations which guarantees that the generation is uniform (every permutation is equally likely).

Table 1 shows the efficiency of the *always theorem* and versions 1.0, 2.0 and 3.0 of our Major Order theorem on sets of permutations of $[n]$, when $n = 15$ or $n = 30$.

Table 1: Efficiency of different approaches, in terms of the proportion of ordering of pairs of elements solved, on sets of uniformly distributed random permutations, n=15 and n=30, $m = 3, 4, 5, 10..15, 20$, statistics generated over 100 000 instances for n=15 and 40 000 instances for n=30.

	$n = 15$				$n = 30$			
m	Always thm	Maj. Order thm 1.0	Maj. Order thm 2.0	Maj. Order thm 3.0	Always thm	Maj. Order thm 1.0	Maj. Order thm 2.0	Maj. Order thm 3.0
3	0.2496	0.3603	0.4981	0.6345	0.2498	0.3055	0.3962	0.5065
4	0.1248	0.2595	0.4711	0.5201	0.1248	0.1937	0.3658	0.4123
5	0.0626	0.1928	0.4648	0.5813	0.0626	0.1272	0.3478	0.4036
10	0.0020	0.0530	0.4435	0.5173	0.0020	0.0199	0.3194	0.3619
11	0.0010	0.0419	0.4478	0.5478	0.0010	0.0139	0.3174	0.3609
12	0.0005	0.0328	0.4418	0.5182	0.0005	0.0098	0.3149	0.3558
13	0.0002	0.0261	0.4457	0.5450	0.0002	0.0070	0.3147	0.3570
14	0.0001	0.0208	0.4415	0.5199	0.0001	0.0050	0.3133	0.3535
15	0.0001	0.0165	0.4453	0.5445	0.0001	0.0036	0.3130	0.3545
20	0	0.0054	0.4415	0.5248	0	0.0007	0.3096	0.3485

A first observation comes from the *always theorem* which solves in average, like stated in Section 3, $\frac{1}{2^{m-1}}$ of the ordering of the pairs in an instance of a set of m uniformly generated random permutations. The *always theorem*, which is englobed in our Major Order theorems, sets an inferior bound on the efficiency of the approach.

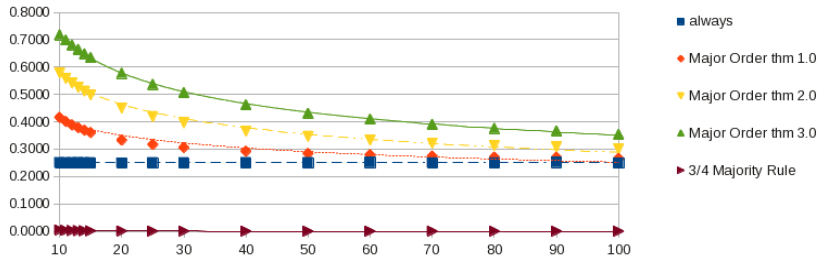
Table 1 also shows that even if our Major Order theorem 1.0 is quite stronger than the *always theorem* on small sets of permutations but quickly converge to

the *always theorem* which keeps a stable proportion of solved pairs as n becomes bigger.

As Table 1 is showing that our Major Order theorems greatly improves the efficiency on small and medium scale instances (in regards to n). The particular strength of versions 2.0 and 3.0 is to extend the efficiency on big set of permutations (bigger m) where the *always theorem* is hardly applicable. More statistics for the Major Order theorems 1.0, 2.0 and 3.0 can be found in Tables 4-5-6 in the Appendix.

Our Major Order theorems have their best efficiency on sets of three permutations ($m = 3$), the only case where the theoretical complexity is still not clear. On a bigger scale, Major Theorem 3.0 can still solve in average more that 33% of the pairs for 3 random permutations of 100 elements. Figure 1 shows the average performance of our approach on 3-permutations sets of different sizes.

Fig. 1: Efficiency of the Always and Major Order Theorems and 3/4 Majority Rule in term of the proportion of pairs resolution, when $m = 3$ and $n = 10..100$, statistics generated over 2000 to 400 000 instances.



To compare our approach with the *3/4 Majority Rule* approach of Betzler *et al.* [2], we first tested its applicability (in terms of finding non-dirty candidates) on uniformly generated sets of m permutations of $[n]$, for different values of m and n. Table 2 shows that this approach is really not doing well on random sets, being applicable less than 1% of the time in most cases. The case where $m = 4$ is an exception, where non-dirty candidates were found in much greater proportions.

In instances with sets of uniformly distributed permutations, we noticed that the vast majority of constraints found by the *3/4 Majority Rule* are also found by the Major Order Theorem 3.0 (see Table 3). Only a few exceptions are found in large scale testing, most of them in instances of the problem with sets of 4 permutations. In approximately 15% of these cases the Major Order theorem 3.0 does not completely englobe the *3/4 Majority Rule*.

5.1 Time complexity of our approach and implementation

We implemented the Major Order Theorems using matrices to represent the constraints. The theoretical complexity for the preprocessing is n^3mk, where n

Table 2: Applicability, in %, of the *3/4 majority rule* on sets of uniformly distributed random permutations, for $n = 8, 9, 10, 15, 20$, $m = 3..10, 15, 20$, statistics generated over 10 000 - 400 000 instances.

$m\backslash n$	8	9	10	15	20
3	0.8%	0.55%	0.41%	0.12%	0.05%
4	16.4%	12.88%	10.37%	3.93%	1.92%
5	2.19%	1.57%	1.16%	0.37%	0.18%
6	0.41%	0.28%	0.2%	0.05%	0.02%
7	0.08%	0.05%	0.03%	0.01%	0%
8	0.88%	0.6%	0.43%	0.12%	0.06%
9	0.22%	0.14%	0.09%	0.02%	0.01%
10	0.05%	0.03%	0.02%	0%	0%
15	0%	0%	0%	0%	0%
20	0%	0%	0%	0%	0%

Table 3: Inclusion, in %, of *3/4 majority rule* in Major Order Thm 3.0 on the same sets as Table 2.

$m\backslash n$	8	9	10	15	20
3	100%	100%	100%	100%	100%
4	85.2%	84.7%	84.0%	86.7%	88.6%
5	100%	100%	100%	99.96%	100%
6	100%	100%	100%	100%	100%
7	100%	100%	100%	100%	100%
8	99.7%	100%	100%	100%	100%
9	100%	100%	100%	100%	100%
10	100%	100%	100%	100%	100%
15	100%	100%	100%	100%	100%
20	100%	100%	100%	100%	100%

is the size of the permutations, m is the number of permutations and k is the number of iterations of the last refined version of the theorem. For each of the $\frac{n(n-1)}{2}$ pair of elements, the construction of collections E_{ij} and E_{ji} implies a scan of the m permutations of A, having each n elements. The cancelations and removals of the elements in those collections are proportional to the scan. Note that in our preliminary tests, k is always really small (between 4 and 9 for sets of permutations of $n = 400$). As an example, on a Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz, computing the Major Order Theorem 3.0 constraints for an instance with $m = 3$ and $n = 400$ take less than 30 seconds.

A simple Branch and Bound (BnB) solver, using some basic left/right constraints [5] and cutting non promising branches, was combined with an implementation of the Major Order Theorem 3.0 to give an evaluation of the computational gain for the computation of a median of a set of permutations. The original BnB solver would take 13 minutes and 53 seconds to solve 1000 uniformly random instances of $m = 3$ and $n = 15$. When combined with our new constraints, the calculation time is reduced to a mere 10 seconds. The source code (Java) is available[2] for testing and replication of the experimental results.

6 Conclusion and future works

In this paper a new approach was presented that partly solve the median problem, under the Kendall-τ distance, by finding a set of ordering constraints on pairs of elements in a median. Its reach is much larger than previous approaches (*always theorem, 3/4 Majority Rule*). Therefore, it is much more efficient on data, especially uniformly-distributed data, which is well-reflected on showed statistics. Our approach has a great efficiency on small and medium scale instances of

[2] http://www-etud.iro.umontreal.ca/ miloszro/caldam/caldam.html

the problem and, curiously, has an even greater impact on cases where $m = 3$. The constraints found by this approach may be used in any algorithm or heuristic to accelerate computations.

It will be interesting to investigate further extensions of the Major Order theorems. Some preliminary results done on cases where any of the multisets E_{ij}, E'_{ij}, ${E''_{ij}}^t$ have a cardinality equal to d_{ij}, are demonstrating a not-negligible improvement of the solving efficiency and a total inclusion of the *3/4-Majority Rule*. The greater efficiency of the Major Order theorems on the particular $m = 3$ case shows great promise for further work. Are there other additional properties, in this particular case, which may further enhance the efficiency?

Acknowledgements. Thanks to Bryan Brancotte, Sarah Cohen-Boulakia and Alain Denise (LRI - Paris Sud) for giving us useful advices and thoughts to guide the work. Thanks to Nicole Burke (Montreal) for a careful english revision of the article.

References

1. N. Ailon, M. Charikar and N. Newman, *Aggregating inconsistent information: ranking and clustering*, Journal of the ACM, 55(5), pp.1–27, 2008.
2. N. Betzler, R. Bredereck, R. Niedermeier, *Theoretical and empirical evaluation of data reduction for exact Kemeny Rank Aggregation*, Autonomous Agents and Multi-Agent Systems, vol. 28, pp.721–748, 2014.
3. N. Betzler *et al.*, *Average parameterization and partial kernelization for computing medians.*, Journal of Computer and System Sciences, 77(4), pp. 774–789, 2011.
4. T. Biedl, F.J. Brandenburg and X. Deng, *Crossings and Permutations*, LNCS 3843, pp. 1–12, 2005.
5. G. Blin, M. Crochemore, S. Hamel and S. Vialette, *Median of an odd number of permutations*, Pure Mathematics and Applications, 21 (2), pp. 161–175, 2011.
6. C. Dwork, R. Kumar, M. Naor and D. Sivakumar, *Rank Aggregation Methods for the Web*, in proceedings of the 10th WWW, pp.613-622, 2001.
7. R.A. Fisher, F. Yates, *Statistical tables for biological, agricultural and medical research*, (3rd ed.). London: Oliver & Boyd. pp. 2627, 1948.
8. M. Karpinski and W. Schudy, *Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament*, LNCS 6506, pp. 3–14, 2010.
9. J. Kemeny, *Mathematics without numbers*, Daedalus 88 , pp. 577591, 1959.
10. M. Kendall, *A New Measure of Rank Correlation*, Biometrika, 30, 81-89, 1938.
11. C. Kenyon-Mathieu and W. Schudy, *How to rank with few errors*, STOC'07, pp. 95-103, 2007.
12. D.E. Knuth, *Seminumerical algorithms*, The Art of Computer Programming 2. Reading, MA: AddisonWesley. pp. 124125, 1969.
13. N. Nishimura and N. Simjour, *Parameterized enumeration of (locally-) optimal aggregations*, IWADS13, LNCS 8037, pp.512–523, 2013.
14. N. Simjour, *Improved parameterized algorithms for the Kemeny aggregation problem*, IWPEC09, LNCS 5917, pp. 312–323, 2009.
15. M. Truchon, *An Extension of the Condorcet Criterion and Kemeny Orders*, Internal Report, Université Laval, 16 pages, 1998.
16. A. vanZuylen and D.P. Williamson, *Deterministic pivoting algorithms for constrained ranking and clustering problems*, Mathematics of Operations Research, 34(3), pp.594–620, 2009.

7 Appendix

Proof of Proposition 1: Let $\mathcal{A}$ be the following set of permutations of $[n]$:

$$\mathcal{A} = \{\sigma_1, \sigma_2, \ldots, \sigma_a, \gamma, \pi_1, \pi_2, \ldots, \pi_a\},$$

where each σ_i, $1 \leq i \leq a$, is a different permutation beginning with element 1 followed by element 2; each π_i, $1 \leq i \leq a$, is a different permutation ending with element 1 followed by element 2, and γ is a permutation beginning by element 2 and ending by element 1 (in γ elements 1 and 2 are thus separated by a set K of k elements). Note: $n = k + 2$.

So, in $2a$ permutations, element 1 is before element 2, and in only one, element 2 is before element 1. Even if the majority order for this pair of elements is 1 before 2, we can show that for an arbitrary a, if $k > 2a - 1$ then the median of $\mathcal{A}$ will be of the form $2K1$, more precisely the median will be $2\alpha 1$ where 2 is before 1 and where α is the optimal permutation of the elements of set K (i.e. $\pi^* = 2\alpha 1$ is a median of $\mathcal{A}$). The idea is that the "pressure" of the set K on elements 1 and 2 will be stronger than the interaction in between 1 and 2. We then choose a minimal a such that $\frac{2a}{2a+1} \geq s$ and $k = 2a$ to complete the construction.

First, we observe that the cost of every element of K is identical in relation to 1 and 2 simply because the elements of K are always grouped together in the permutations of $\mathcal{A}$. So, in a certain permutation of $\mathcal{A}$, if an element $x \in K$ is on the left (resp. on the right) of 1 then all other elements of K will be on the left (resp. on the right) of 1. It goes the same way for the left/right of 2 and it applies in all permutations of $\mathcal{A}$.

Second, we can state that there exist at least one optimal arrangement for the elements of K based on the order they were placed in the permutations of $\mathcal{A}$. Let α be any optimal arrangement for the elements of K and C_α, its associated cost.

The two previous points are leading to this observation: a median permutation of $\mathcal{A}$ will have the elements of K arranged following α. This is because of the fact that if a supposed median permutation have the elements of K arranged in an non-optimal way, we could make a bunch of swaps between those elements of K, omitting 1 and 2, so that their relative order will become an optimal arrangement, thus lowering the cost of the permutation.

Now, a median of $\mathcal{A}$ can either be of the form $K_1 1 K_2 2 K_3$ or of the form $K_1 2 K_2 1 K_3$, where the K_i are possibly empty subsets of K, such that $\bigcup_{i=1}^{3} K_i = K$, $K_i \bigcap K_j = \emptyset$ if $i \neq j$, $|K_i| = k_i$ and $k_1 + k_2 + k_3 = k$. The order of the elements of K is known and follows α, so the only variables to be set are the numbers of elements to be placed in K_1, K_2 and K_3. In other words, we have α and we need to place in 1 and 2 in an optimal way to attain the median.

If our median is of the form $K_1 1 K_2 2 K_3$ we have

$$\begin{aligned} d_{KT}(K_1 1 K_2 2 K_3, \mathcal{A}) &= \overbrace{a(2k_1 + k_2)}^{1} + \overbrace{a(k_2 + 2k_3)}^{2} + \overbrace{1(1 + k_1 + 2k_2 + k_3)}^{3} + C_\alpha \\ &= 2ak + 1 + k_1 + 2k_2 + k_3 + C_\alpha, \end{aligned}$$

where terms 1, 2 and 3 capture the distance with respectively the permutations σ_i, $1 \leq i \leq a$, the permutations π_i, $1 \leq i \leq a$, and the permutation γ of $\mathcal{A}$, for the pairs $(1, x)$, $(2, x)$ and $(1, 2)$, $x \in K$. The cost for the pairs (x_1, x_2), $x_1, x_2 \in K$, is counted in C_α.

Following the same logic, if our median is of the form $K_1 2 K_2 1 K_3$, we have

$$\begin{aligned} d_{KT}(K_1 2 K_2 1 K_3, \mathcal{A}) &= a(2k_1 + k_2 + 1) + a(k_2 + 2k_3 + 1) + 1(k_1 + k_3) + C_\alpha \\ &= 2ak + 2a + k_1 + k_3 + C_\alpha. \end{aligned}$$

If we try to minimize $d_{KT}(K_1 1 K_2 2 K_3, \mathcal{A})$, we will have to put all the k elements of K in either K_1 or K_3 because putting them in K_2 will have a doubled penalty for the cost. Doing so, the minimal cost for $K_1 1 K_2 2 K_3$ is $2ak+1+k+C_\alpha$. Now, if we try to minimize $d_{KT}(K_1 2 K_2 1 K_3, \mathcal{A})$, we will have to put all the k elements of K in K_2 because putting them in either K_1 or K_3 will give a penalty whereas K_2 is "silent" in the cost function. So, the minimal cost for $K_1 2 K_2 1 K_3$ is $2ak + 2a + C_\alpha$. Since it is stated in the proposition that $k > 2a - 1$, the minimal cost of $K_1 1 K_2 2 K_3$ is strictly greater that the minimal cost of $K_1 2 K_2 1 K_3$ ($2ak + 1 + k + C_\alpha > 2ak + 2a + C_\alpha$).

This gives us that median permutations π^* of $\mathcal{A}$ are of the general form $K_1 2 K_2 1 K_3$, where K_1 and K_3 are empty. Thus π^* is of the form $2K1$ or more precisely $\pi^* = 2\alpha 1$, where α be any optimal arrangement for the elements of K with respect to $\mathcal{A}$. Element 2 is favored to element 1 in the median, regardless of the big proportion of permutations of $\mathcal{A}$ favoring 1 to 2, which finish the proof of non-existence of a %-majority bound. ■

Other tables:

Table 4: Efficiency of the Major Order theorem 1.0 on sets of uniformly distributed random permutations, from $n = 8$ to $n = 100$, $m = 3$ to $m = 50$, statistics generated over 100 000 instances for smaller n to 2000 instances for bigger n (see Table 7)

$m \backslash n$	8	10	15	20	25	30	40	50	60	80	100
3	0.4560	0.4155	0.3603	0.3339	0.3170	0.3056	0.2914	0.2840	0.2785	0.2712	0.2667
4	0.3672	0.3224	0.2595	0.2274	0.2071	0.1939	0.1768	0.1673	0.1608	0.1514	0.1470
5	0.3076	0.2591	0.1928	0.1597	0.1403	0.1272	0.1110	0.1011	0.0943	0.0859	0.0823
10	0.1449	0.1004	0.0530	0.0344	0.0251	0.0198	0.0139	0.0108	0.0089	0.0068	0.0058
15	0.0734	0.0432	0.0165	0.0085	0.0052	0.0036	0.0020	0.0014	0.0010	0.0006	0.0005
20	0.0387	0.0193	0.0054	0.0022	0.0011	0.0007	0.0003	0.0002	0.0001	0.0001	0.0000
25	0.0209	0.0088	0.0018	0.0006	0.0003	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
30	0.0114	0.0041	0.0006	0.0002	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
35	0.0063	0.0020	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	0.0035	0.0009	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
45	0.0020	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.0011	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 5: Efficiency of the Major Order theorem 2.0 on sets of uniformly distributed random permutations, from $n = 8$ to $n = 100$, $m = 3$ to $m = 50$, statistics generated over 100 000 instances for smaller n to 2000 instances for bigger n (see Table 7)

$m\backslash n$	8	10	15	20	25	30	40	50	60	80	100
3	0.6292	0.5784	0.4981	0.4513	0.4194	0.3962	0.3651	0.3462	0.3324	0.3139	0.3020
4	0.5652	0.5333	0.4711	0.4265	0.3923	0.3661	0.3268	0.3003	0.2801	0.2504	0.2320
5	0.5966	0.5469	0.4648	0.4131	0.3761	0.3479	0.3073	0.2781	0.2555	0.2232	0.2033
10	0.5730	0.5253	0.4435	0.3886	0.3493	0.3193	0.2749	0.2439	0.2197	0.1878	0.1656
15	0.5929	0.5377	0.4453	0.3863	0.3448	0.3129	0.2679	0.2361	0.2122	0.1791	0.1561
20	0.5820	0.5310	0.4415	0.3832	0.3415	0.3097	0.2647	0.2323	0.2089	0.1750	0.1523
25	0.5979	0.5396	0.4444	0.3842	0.3415	0.3099	0.2625	0.2317	0.2074	0.1736	0.1500
30	0.5878	0.5349	0.4426	0.3828	0.3403	0.3082	0.2616	0.2293	0.2055	0.1715	0.1498
35	0.5996	0.5419	0.4455	0.3843	0.3407	0.3083	0.2612	0.2285	0.2050	0.1711	0.1481
40	0.5922	0.5374	0.4439	0.3828	0.3402	0.3070	0.2603	0.2288	0.2038	0.1707	0.1483
45	0.6016	0.5435	0.4461	0.3848	0.3405	0.3084	0.2615	0.2278	0.2038	0.1708	0.1472
50	0.5956	0.5395	0.4440	0.3833	0.3402	0.3072	0.2600	0.2284	0.2037	0.1700	0.1467

Table 6: Efficiency of the Major Order theorem 3.0 on sets of uniformly distributed random permutations, from $n = 8$ to $n = 100$, $m = 3$ to $m = 50$, statistics generated over 100 000 instances for smaller n to 2000 instances for bigger n (see Table 7)

$m\backslash n$	8	10	15	20	25	30	40	50	60	80	100
3	0.7642	0.7179	0.6345	0.5792	0.5378	0.5059	0.4608	0.4312	0.4086	0.3775	0.3563
4	0.6010	0.5757	0.5201	0.4761	0.4406	0.4127	0.3700	0.3404	0.3176	0.2833	0.2612
5	0.7381	0.6853	0.5813	0.5046	0.4470	0.4038	0.3456	0.3073	0.2793	0.2413	0.2191
10	0.6440	0.6019	0.5173	0.4515	0.4012	0.3617	0.3040	0.2649	0.2354	0.1979	0.1726
15	0.7392	0.6732	0.5445	0.4581	0.3987	0.3544	0.2946	0.2543	0.2254	0.1869	0.1612
20	0.6739	0.6262	0.5248	0.4484	0.3917	0.3486	0.2896	0.2491	0.2209	0.1819	0.1566
25	0.7463	0.6774	0.5440	0.4557	0.3939	0.3497	0.2867	0.2480	0.2186	0.1800	0.1540
30	0.6899	0.6391	0.5307	0.4496	0.3904	0.3464	0.2852	0.2448	0.2164	0.1775	0.1536
35	0.7490	0.6804	0.5467	0.4563	0.3928	0.3474	0.2846	0.2438	0.2156	0.1770	0.1517
40	0.7008	0.6473	0.5349	0.4505	0.3905	0.3448	0.2833	0.2438	0.2142	0.1764	0.1518
45	0.7522	0.6836	0.5484	0.4570	0.3924	0.3473	0.2847	0.2427	0.2141	0.1764	0.1507
50	0.7095	0.6533	0.5367	0.4518	0.3908	0.3450	0.2827	0.2431	0.2138	0.1755	0.1501

Table 7: Number of instances generated and calculated for each couple $m\backslash n$ (depending only on n)

n	8	10	15	20	25	30	40	50	60	80	100
# instances	100000	100000	100000	100000	100000	50000	20000	10000	8000	4000	2000

CHAPITRE 4

CAS INVARIANT A=M(A)

4.1 Résumé

L'article propose une étude du cas invariant où l'ensemble de permutations médianes de A est identique à A. On y montre deux cas possibles d'instances de A pour lesquels A=M(A) avec des théorèmes à l'appui.

L'objectif est de classifier les instances du problème qui sont plus faciles à résoudre que le problème général qui est NP-complet. Ici, les instances de la classe A=M(A) se résolvent en temps polynomial.

L'extended abstract "Medians of permutations : when A = M(A)" a été accepté à la conférence International Permutation Patterns Conference, July 2014, Johnson City, TN, USA (IPPC2014) et se retrouve dans les proceedings de celle-ci [18].

4.2 Partage du travail

Dans cet article, j'ai développé les deux théorèmes présentés : cas circulaire et cas S_k. Sylvie Hamel et moi avons rédigé les preuves. Sylvie Hamel a rédigé l'article.

Medians of permutations: when $A = \mathcal{M}(A)$ *

Sylvie Hamel and Robin Milosz
DIRO - Université de Montréal - QC, Canada

June 2, 2016

Abstract

Let $A = \{\pi_1, \pi_2, \ldots, \pi_m\}$ be a set of m permutations of $\{1, 2, \ldots, n\}$. The set of medians of A, denoted $\mathcal{M}(A)$, is the set of all permutations in $\mathcal{S}_n$ that are the "closest" of A under a distance fucntion d. Here we investigated the case $A = \mathcal{M}(A)$ under the Kendall-τ distance.

1 Introduction

The problem of finding medians of a set of m permutations of $\{1, 2, \ldots, n\}$ under the Kendall-τ distance [8, 12] is often cited in the literature as the Kemeny Score Problem [7]. In this problem m voters have to order a list of n candidates according to their preferences. The problem then consist in finding a *Kemeny consensus*; an order of the candidates that agrees the most with the order of the m voters, *i.e.*, that minimizes the sum of the disagreements.

This problem is polynomial-time solvable for $m = 2$, has been proved to be NP-complete when $m \geq 4$ [6] but its complexity remains unknown for $m = 3$. In the last 10 years, different approximation algorithms have been derived [1, 2, 3, 5, 9, 13]. In [4] we began to study the problem in a more combinatorial point a view, deriving some nice properties of the medians of a set of m permutations, with m odd.

In this extended abstract, we give a first investigation of the case $A = \mathcal{M}(A)$ i.e. we want to know exactly when a set of permutations is equal to the set of its medians. The abstract is organized as follow. After introducing the basic definitions and notations (in Section 2), we provide, in Section 3, some special cases of sets of permutations A for which $A = \mathcal{M}(A)$. In Section 4, we conclude and give some ideas to completely solve this $A = \mathcal{M}(A)$ case.

2 Definitions and notations

A **permutation** π is a bijection of $[n] = \{1, 2 \ldots, n\}$ onto itself. The set of all permutations of $[n]$ is denoted $\mathcal{S}_n$. As usual we denote a permutation π of $[n]$ as

*partly supported by NSERC through an Individual Discovery Grant (Hamel)

$\pi = \pi_1\pi_2 \dots \pi_n$. The **identity permutation** correspond to the identity bijection of $[n]$ and is denoted $\imath = 12 \dots n$. A pair (π_i, π_j) of elements of the permutation π is called an **inversion** if $\pi_i > \pi_j$ and $i < j$. The **number of inversion** of a permutation π is denoted $\text{inv}(\pi)$.
The **Kendall-τ distance**, denoted d_{KT}, counts the number of pairwise disagreements between two permutations and can be defined formally as follows: for permutations π and σ of $[n]$, we have that

$$d_{KT}(\pi, \sigma) = |(i, j) : i < j \text{ and } [(\pi[i] < \pi[j] \text{ and } \sigma[i] > \sigma[j]) \text{ or } (\pi[i] > \pi[j] \text{ and } \sigma[i] < \sigma[j])|,$$

where $\pi[i]$ denote the position of integer i in permutation π. Note that we can easily computed $\text{inv}(\pi)$ as $\text{inv}(\pi) = d_{KT}(\pi, \imath)$.

The **problem of finding $\mathcal{M}(A)$, the set of all medians of A under the Kendall-τ distance** can be stated formally as follow:

Given $A \subseteq \mathcal{S}_n$, we want to find all permutations π^* of $[n]$ such that $$d_{KT}(\pi^*, A) \leq d_{KT}(\pi, A), \text{ for all } \pi \in \mathcal{S}_n.$$

3 The case $A = \mathcal{M}(A)$

In this section, we give two special cases when the set of medians of A is equal to A itself. The first case (see Theorem 1) is when A is composed of a permutation and its "circular rotations". The second one (see Theorem 2) is when A consist of permutations composed of a common S_k kernel with some (or none) shared fixed points.

Theorem 1 *Let $\pi = \pi_1 \dots \pi_n$ be a permutation of $[n]$ and let $\uparrow_i \pi = \pi_{i+1} \dots \pi_n\pi_1 \dots \pi_i$. If $A = \{\uparrow_i \pi \mid 0 \leq i \leq n-1\}$ then $A = \mathcal{M}(A)$ and, $\forall i$, $0 \leq i \leq n-1$, $d_{KT}(\uparrow_i \pi, A) = \frac{(n+1)n(n-1)}{6}$.*

Proof. We have that the set A contains the following permutations:

$$\begin{array}{ccccccc}
\pi_1 & \pi_2 & \pi_3 & \dots & \pi_{n-2} & \pi_{n-1} & \pi_n \\
\pi_2 & \pi_3 & \pi_4 & \dots & \pi_{n-1} & \pi_n & \pi_1 \\
\pi_3 & \pi_4 & \pi_5 & \dots & \pi_n & \pi_1 & \pi_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\pi_n & \pi_1 & \pi_2 & \dots & \pi_{n-3} & \pi_{n-2} & \pi_{n-1}
\end{array}$$

Now to see that $d_{KT}(\uparrow_i \pi, A) = (n+1)n(n-1)/6$ just start with $\uparrow_i \pi$ for any i and counts the number of disagreements with all permutations in set A. For any ℓ,

$1 \leq \ell \leq n-1$, any of the $n-\ell$ pairs of elements π_ℓ and $\pi_{\ell+k}$, $1 \leq k \leq n-\ell$, in $\uparrow_i \pi$ appears in the order π_ℓ before $\pi_{\ell+k}$ in all permutations except the ℓ permutations of A having respectively $\pi_\ell, \pi_{\ell+1}, \ldots \pi_{\ell+k-1}$ as its last element. So we have

$$d_{KT}(\uparrow_i \pi, A) = \sum_{\ell=1}^{n} (n-\ell)\ell = \frac{(n-1)n(n+1)}{6}.$$

To finish the proof, note that there are no pairs of elements that appears in the same order in all permutations of A. That means that any pairs of elements of a permutation σ will contribute with a strictly positive factor to the distance $d_{KT}(\sigma, A)$. Now, the permutations in A are the only one in $\mathcal{S}_n$ that have $n-1$ pairs of element (all consecutive pairs π_i, π_{i+1}) that contribute only by 1 to the distance. That means that for any other permutation $\sigma \in \mathcal{S}_n \setminus A$, we only have $t < n-1$ pairs of elements that contribute by 1 to the distance. All the order pairs contribute by a factor $f > 1$. We can then conclude that $d_{KT}(\sigma, A) > d_{KT}(\uparrow_i \pi, A), \forall i$, which means that $A = \mathcal{M}(A)$. ■

Theorem 2 *If* $A = \{1\,2 \ldots \ell\ \sigma\ \ell{+}k{+}1 \ldots n \quad | \quad \sigma$ *any permutation of* $\{\ell{+}1, \ldots, \ell{+}k\}\}$ *for* $0 \leq \ell \leq n-1$, $1 \leq k \leq n$, *then* $A = \mathcal{M}(A)$ *and for any* $\pi \in A$, $d_{KT}(\pi, A) = \frac{k(k-1)k!}{4}$. *Note that* $A = \mathcal{S}_n$, *is the special case when* $\ell = 0$ *and* $k = n$.

Proof. For any permutation $\gamma \in \mathcal{S}_n$, $d_{KT}(\gamma, A)$ contains the distance to $\mathcal{S}_k$. This distance is the total number of inversions in all permutations of $[k]$, which is $\frac{k(k-1)k!}{4}$ [10, 11]. Since for any permutation $\in A$, the other elements are fixed points, $d_{KT}(\pi, A) = \frac{k(k-1)k!}{4}$, and these element represent the elements of $\mathcal{S}_n$ with minimal Kendall-τ distance from A. ■

4 Conclusion and future works

Since the problem of finding medians of a set of permutation under the Kendall-τ distance is NP-complete in general, finding special sets A for which it is easy to find the medians is useful for computational purposes. In this extended abstract we have just begin to study the special case $A = \mathcal{M}(A)$ but there are still a lot of interesting ideas to investigate. The cases $A \subset \mathcal{M}(A)$, $\mathcal{M}(A) \subset A$ and $A \cap \mathcal{M}(A) = \emptyset$ are only three of some interesting future directions that needs to be evaluate.

Coming back to our present case, the two theorems presented in Section 3 gives us the basis for building further theory on the $A = \mathcal{M}(A)$ problem. If we extend Theorem 2 to a composition of basic structures (circular types, as in Theorem 1, and $\mathcal{S}_k$ types), we conjecture that this type of composition keeps the $A = \mathcal{M}(A)$ property. With this extension, we can rethink the permutations of A, in Theorem 2, as a composition of a S_k structure with some or none S_1 structures (the fixed points).

We observed that some special permutation sets display the $A = \mathcal{M}(A)$ property by addition of basic structures instead of composition. It would be interesting to

verify if there exist a relation between circular and $\mathcal{S}_k$ structures, knowing that a $\mathcal{S}_k$ structure can be viewed as an addition of circular components.

For now, we leave you with this question, that we plan to answer in the near future: Can we characterize all the $A = \mathcal{M}(A)$ cases?

References

[1] N. Ailon, M. Charikar and N. Newman, *Aggregating inconsistent information: Ranking and clustering*, In Proceedings of the 37th STOC, pp.684–693, 2005.

[2] N. Betzler, M.R. Fellows, J. Guo, R. Niedermeier and F.A. Rosamond, *Fixed-Parameter Algorithms for Kemeny Scores*, LNCS 5034, pages 60–71, 2008.

[3] T. Biedl, F.J. Brandenburg and X. Deng, *Crossings and Permutations*, LNCS 3843, pages 1–12, 2005.

[4] G. Blin, M. Crochemore, S. Hamel et S. Vialette, *Medians of an odd number of permutations*, Pure Mathematics and Applications, Volume 21, Issue 2, pp. 161–175, 2010.

[5] V. Conitzer, A. Davenport and J. Kalagnanam, *Improved Bounds for Computing Kemeny Rankings*, in Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), pages 620–627, 2006.

[6] C. Dwork, R. Kumar, M. naor and D. Sivakumar, *Rank Aggregation Methods for the Web*, in proceedings of the 10th WWW, pp.613-622, 2001.

[7] J. Kemeny, *Mathematics without numbers*, Daedalus 88 , pp. 577591, 1959.

[8] M. Kendall, *A New Measure of Rank Correlation*, Biometrika, 30, 81-89, 1938.

[9] C. Kenyon-Mathieu and W. Schudy, *How to rank with few errors*, In Proceedings of the 39th STOC, pp. 95-103, 2007.

[10] M. Stern, *Aufgaben*, Jour. für reine und agrew. Math., 18, pp. 100, 1838.

[11] M. O. Terquem, *Solution d'un Problème de combinaison*, J. de Math. Pures et Appl., 3, pp. 559-560, 1838.

[12] M. Truchon, *An Extension of the Condorcet Criterion and Kemeny Orders*, Internal Report, cahier 98-15 du Centre de Recherche en Économie et Finance Appliquées, Université Laval, 16 pages, 1998.

[13] A. vanZuylen and D.P. Williamson, *Deterministic algorithms for rank aggregation and other ranking and clustering problems*, in Proceedings of the 5th WAOA, LNCS 4927, pp.260–273, 2007.

CHAPITRE 5

GÉNÉRALISATION AUX PERMUTATIONS AVEC ÉGALITÉS

5.1 Définition du problème généralisé

Un classement est une permutation dans laquelle on introduit la relation d'égalité entre deux éléments qui sont alors placés à la même position, c.-à-d. dans le même "panier". La relation reste transitive et l'ordre pour chaque paire est défini. L'ensemble de tous les classements de taille n est noté $Rank_n$.

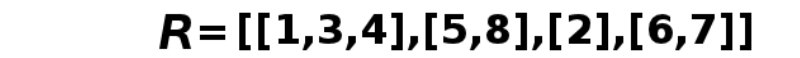


Figure 5.1 – Un exemple d'un classement. Ici, il y a 8 éléments dans 4 paniers. Les éléments 5 et 8 sont à égalité ($5 = 8$) alors que l'élément 5 est avant l'élément 2 ($5 \prec 2$). Par transitivité (aussi par observation), on a que $8 \prec 2$.

On définit la distance de Kendall-τ généralisée entre deux classements comme le nombre de paires qui ne partagent pas le même ordre dans les deux classements, c'est-à-dire le nombre de paires en désaccord :

$$\begin{aligned} K^{(p)}(R,T) \;=\; & \#\{(i,j) : i<j \quad \wedge [(R[i]<R[j] \wedge T[i]>T[j]), \vee \\ & \qquad\qquad (R[i]>R[j] \wedge T[i]<T[j]))\} \\ & + p \times \#\{(i,j) : i<j \quad \wedge (R[i]=R[j] \wedge T[i]\neq T[j]), \vee \\ & \qquad\qquad (R[i]\neq R[j] \wedge T[i]=T[j])]\} \end{aligned}$$

La variable $p \in [0,1]$ permet de pénaliser plus ou moins les désaccords liées aux égalités selon le problème considéré. Dans le cas étudié ici, on sera intéressé par un coût de $p = 1$, c'est-à-dire qu'on pénalisera ces désaccords au même niveau que les désaccords habituels. Pour simplifier la notation, on utilisera $K(R,T)$ à la place de $K^{(p)}(R,T)$. Comme pour le cas simple, la distance d'un classement à un ensemble de classements

est la somme des distances entre le classement et chaque classement de l'ensemble.

$$K(R,\mathscr{R}) = \sum_{R_i \in \mathscr{R}} K(R,R_i)$$

Le problème de la médiane de classements se définit mathématiquement comme suit : Soit un ensemble de classements $\mathscr{R} = \{R_1, \ldots R_t\}$, $R_i \in Rank_n$, trouver l'ensemble des classements $R^* \in Rank_n$ tel que :

$$K(R^*,\mathscr{R}) \leq K(R,\mathscr{R}), \forall R \in Rank_n.$$

Cet ensemble de classements médians est dénoté $M(\mathscr{R})$.

La difficulté supplémentaire du problème est partiellement liée à la taille de l'espace des classements $Rank_n$ qui est plus grand que l'espace des permutations S_n (voir Tableau 5.I). En fait, S_n est un sous-ensemble de $Rank_n$. La formule pour calculer la taille de $Rank_n$ s'obtient de façon récursive. Un classement de n éléments peut commencer avec un premier panier de taille i, i allant de 1 jusqu'à n. Pour une taille de panier i fixée, il y a $\binom{n}{i}$ façons de choisir les i éléments qui vont entrer dans le premier panier. Après le choix d'éléments, il reste $n-i$ éléments à placer après ce premier panier, ce qui revient au nombre possible de classements avec $n-i$ éléments. Il n'y a qu'une façon de placer un seul élément et par définition, il y a aussi qu'une façon de ne rien placer du tout. Le nombre de classements possibles (ou $\#Rank_n$) peut être calculé par la récurrence suivante :

$$f(0) = f(1) = 1$$

$$f(n) = \sum_{i=1}^{n} \binom{n}{i} f(n-i)$$

$$\#Rank_n = f(n)$$

Je tiens à remercier Charles Desharnais avec qui j'ai développé une formule d'approximation $|Rank_n| \approx \frac{n!}{2(\log 2)^{n+1}}$ avant de nous apercevoir, après une recherche sur l'"Online Encyclopedia of Integer Sequences" (OEIS)[1], que cette formule a déjà été découverte.

[1]La Online Encyclopedia of Integer Sequences, http://oeis.org/, est un site qui regroupe toutes les

n	$\#S_n$	$\#Rank_n$	$\frac{n!}{2(\log 2)^{n+1}}$	ratio $\frac{\#Rank_n}{\#S_n}$
1	1	1	1.041	1.000
2	2	3	3.003	1.500
3	6	13	12.996	2.166
4	24	75	74.999	3.125
5	120	541	541.002	4.508
6	720	4 683	4 683.001	6.504
7	5 040	47 293	47 292.999	9.384
8	40 320	545 835	545 834.998	13.538
9	362 880	7 087 261	7 087 261.002	19.531

Tableau 5.I – Cardinalités des espaces S_n et $Rank_n$, de l'approximation pour $\#Rank_n$ et ratio entre les tailles de ces espaces.

Le site OEIS nous donne deux résultats pour cette suite : A000670 "Fubini numbers", A034172 "Nearest integer to $n!/(2 * \log(2)^{n+1})$", cette dernière formule correspond à notre approximation.

La complexité du problème généralisé reste NP-complète, car le problème simple y est une sous-classe. De plus, il est évident que si une instance du problème général ne contient aucune égalité, juste des paniers avec un seul élément, alors la solution n'en contiendra pas non plus. Ce problème revient au cas simple dont on connaît la complexité : NP-Complet.

5.2 Travaux précédents

Le problème de la médiane de classement est connu dans la littérature mais moins de travaux ont été effectués sur le sujet. [33] et [1] se sont concentré sur l'obtention d'un ordre des p-premiers candidats avec des heuristiques. Dans [5], un extension du cas simple de l'approche à paramètres fixes est étudiée. En 2011, une heuristique et une réduction de données sont proposées dans [10], puis appliqués sur des données biologiques (qu'on va réutiliser plus loin dans ce chapitre). Cela a été suivi en application réele en 2014 par le serveur web "ConQuR-Bio"[4], basé sur l'heuristique de [10], qui ordonne

séquences d'entiers connues et les recherches publiées sur celles-ci.

des résultats provenant de plusieurs classements pour une requête dans le domaine de la bioinformatique.

5.3 Contexte et applications

Dans la pratique, il n'est pas toujours possible de différencier des candidats similaires, il y a alors des éléments à égalité dans ces classements.

Une application en bio-informatique est le classement d'un ensemble de gènes liés à une maladie en différents groupements d'importance allant des plus importants au moins important. Comme différentes méthodes classent ces gènes dans des groupements différents, la recherche d'un consensus de ces différents classements est importante pour faire ressortir les gènes classés intéressants par plusieurs méthodes. (Voir Figure 5.3)

Dans ces cas décrits, l'utilisation de permutations généralisées (ou classement) est plus appropriée pour traiter le problème et trouver un consensus.

5.4 Algorithme et heuristique

L'implémentation de l'algorithme exact du B&B pour cette généralisation n'a pas été aussi fructueuse que pour le cas simple, car l'espace de recherche est plus grand et qu'il est plus difficile de déduire des contraintes pour nous aider dans la recherche de consensus. À titre indicatif, ce B&B prend en moyenne plus de 4000ms pour résoudre une instance de $m = 3$ et $n = 15$ alors que dans le cas simple le B&B prend en moyenne 10ms pour la même taille d'instance. Seulement la contrainte **borneInf + semiDist** a été implémentée dans notre version du B&B pour ce problème (voir chapitre 2.3.1). Il reste certainement plusieurs améliorations possibles telles que débuter le B&B avec une approximation de SA ou prendre en compte les contraintes décrites par l'article présenté à la fin de cette section. Ce sont deux astuces déjà implémentées dans le cas simple de permutations et décrit dans la section 2.3.1.

Par contre, une implémentation de l'heuristique SA s'est avérée aussi efficace sur cette généralisation du problème avec l'ajout d'un paramètre de fission/fusion α pour

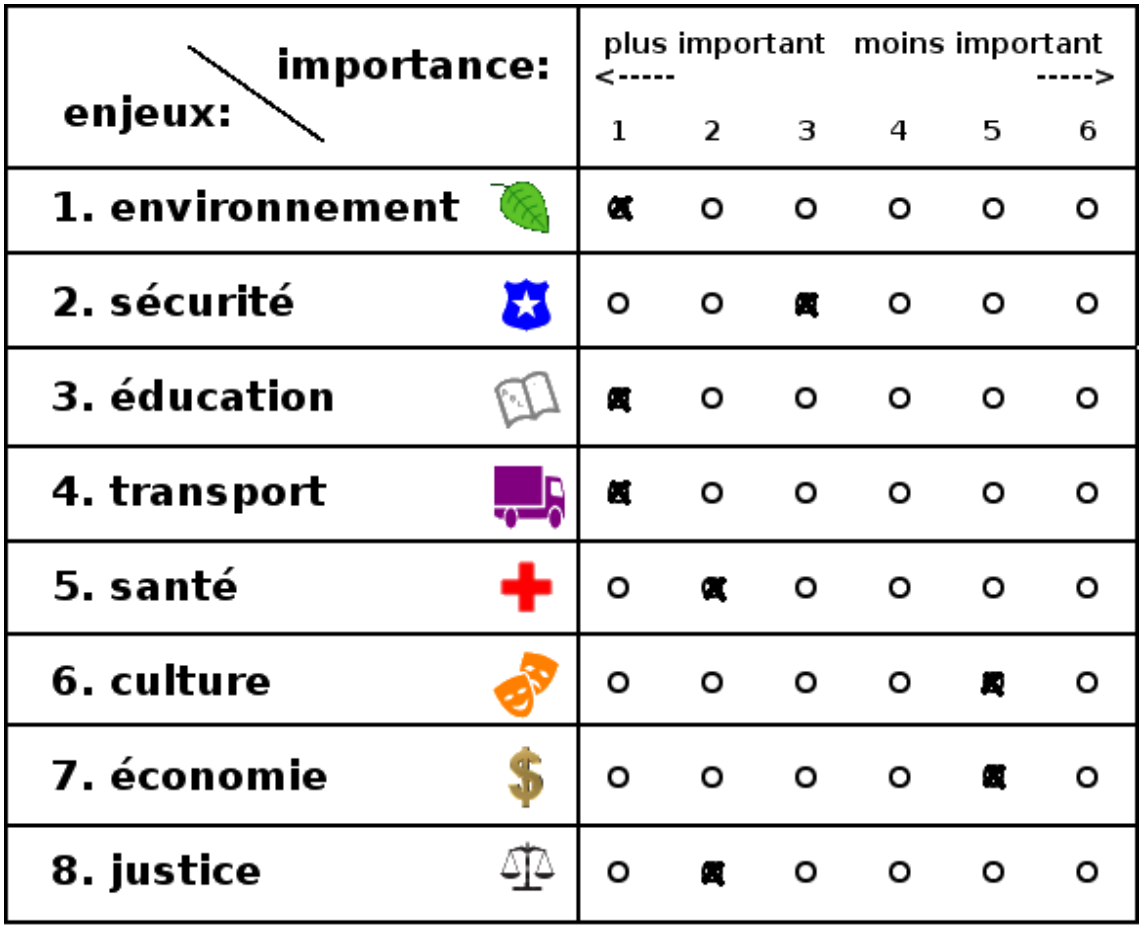


Figure 5.2 – Exemple de questionnaire qui engendre des classements pour les choix des politiques. Ce type de questionnaire est très populaire en sciences sociales et nous est assez familier. Un remplissage correct de ce questionnaire peut être interprété comme un classement avec égalités.

les mouvements générés aléatoirement. On se rappelle de la section 2.2, qu'une méthode a été proposée pour générer une solution alternative par un mouvement circulaire (voir Figure 2.2). Le mouvement circulaire est aussi utilisé dans le cas présent, par contre il faut tenir compte qu'ici des éléments peuvent être à égalité i.e. dans le même panier. La

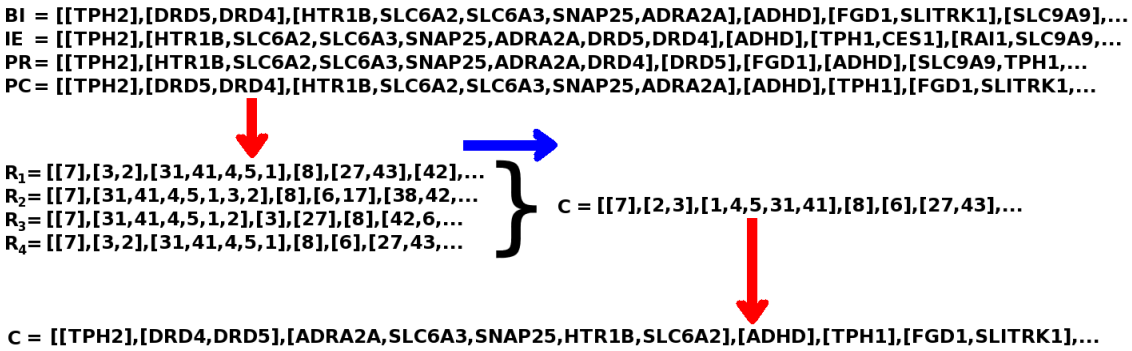


Figure 5.3 – Exemple de classements de gènes. Dans cette figure, quatre classements de gènes liés à la maladie de l'ADHD (Attention Deficit Hyperactivity Disorder) sont présentés dans des groupements ordonnés selon l'importance. On modélise le problème en associant un entier à chaque geène et on trouve le consensus de ces classements. On obtient alors un classement qui fait ressortir les gènes importants. On peut ensuite se baser sur ce classement pour orienter des futures recherches. Cet exemple provient de données utilisées dans la section 5.6. Méthodes de classement : BI=Bioggle, IE=In Edge, PR=Page Rank et PC=Path Count.

perturbation pour obtenir une solution alternative fonctionne comme suit : un élément aléatoire est sélectionné, puis une valeur aléatoire uniforme $x \in [0,1]$ est choisie, si cette valeur est en dessous du paramètre de fission/fusion $x < \alpha$ alors le mouvement sera un mouvement de fission sinon ($x \geq \alpha$) ce sera un mouvement de fusion. Un mouvement de fission consiste à prendre l'élément sélectionné et le placer dans un nouveau panier. À l'opposé, un mouvement de fusion consiste à prendre l'élément sélectionné et le placer dans un autre panier existant. Le paramètre de fission/fusion a été établi à $\alpha = 0.6$ pour les cas de l'application sur des données bio-informatique. L'implémentation ressemble beaucoup à celle du cas simple. Le pseudo-code de l'heuristique SA pour le cas généralisé se retrouve à Algorithm 3 dans l'annexe II. Une étude sur l'efficacité du SA sur des données biologiques est présentée plus tard dans la section 5.6.

5.5 Compressions et réduction de données

L'article présenté à la fin de ce chapitre offre un théorème qui prouve que si deux éléments sont toujours à égalité dans les classements, alors ils vont être toujours à égalité dans les classements médians. Il est donc plus avantageux de considérer ces éléments en même temps et de les placer/déplacer ensemble dans les approches algorithmiques

et heuristiques. On introduit alors la notion de compression exacte de données où les éléments mis ensemble ont été prouvé théoriquement d'être à égalité dans les solutions optimales. Plus loin dans cette section, nous allons introduire la notion de compression majoritaire (heuristique) où les éléments mis ensemble ont une grande probabilité de se retrouver à égalité dans une solution optimale.

La compression exacte de données consiste en un prétraitement de l'instance de départ. On analyse les classements de notre ensemble de départ et on groupe les éléments qui sont toujours ensemble. Évidemment, la relation est transitive c.-à-d. si i est toujours à égalité avec j et que j est toujours à égalité avec k alors i est toujours à égalité avec k. Chaque groupe d'éléments devient un unique élément lourd (le poids étant le nombre d'éléments associés). Les matrices de coût sont ajustées en conséquence. Nous allons renommer les éléments pour avoir une numérotation consistante $\{1,2,3,\ldots n_c\}$ (voir Figure 5.4). On note que la compression exacte réduit la taille des classements considérés de n à n_c $(n \rightarrow n_c)$.

Cette compression de données est une réduction de données (data reduction) qui permet non seulement d'accélérer les calculs computationnels, car il y a simplement moins d'éléments à traiter, mais aussi de rendre le problème plus facile. En effet, on se limite au sous-espace de recherche qui contient toujours les éléments groupés ensemble dans les mêmes paniers. Sans compression, déplacer un groupe d'éléments toujours à égalité devrait se traduire en une série de mouvements consécutifs de ces éléments (voir Figure 5.5) qui auraient des scores moins bons (donc moins probable pour SA).

Finalement, des travaux préliminaires ont été faits sur des compressions plus fortes, où l'on essai de réduire davantage la taille des données. Cette idée a été inspirée des observations de données biologiques. À titre indicatif, une méthode consiste à regrouper non seulement les éléments toujours à égalité, mais aussi les éléments majoritairement à égalité. Dans ce cas, on recherche des sous-graphes complets K_n dont chaque élément est majoritairement à égalité avec chacun des autres éléments. Ces sous-graphes formeront des éléments lourds. Plusieurs astuces additionnelles permettent de gérer les cas rares ou dégénérés qui ne fonctionnent pas bien avec cette compression heuristique. Ces compressions majoritaires permettent une réduction plus forte des données. Pour

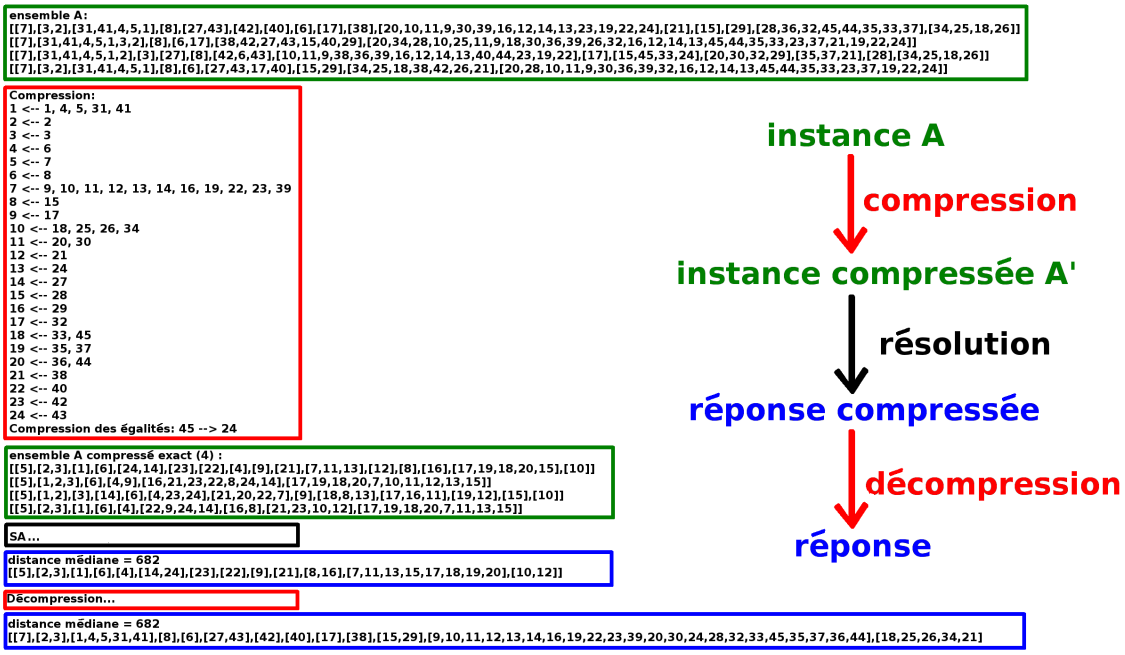


Figure 5.4 – Compression exacte. Cette figure illustre un exemple en application sur un cas de données bio-informatiques. On commence avec l'instance de départ, on effectue la réduction de données par compression exacte, on trouve le consensus, on effectue la décompression et on a la solution pour le problème d'origine. On remarque la différence importante de taille des classements. Pour les détails de cet exemple, voir l'annexe II. C'est le cas ADHD de nos données (Tableau 5.II, "cas A").

l'exemple de la Figure 5.4 où la compression exacte permet de compresser la taille originale des classements d'un taille originale de 45 à une taille de 24 ($45 \rightarrow 24$), une application des compressions majoritaires donne une meilleure compression passant de 45 à 19 ($45 \rightarrow 19$). De plus, le cas (non présenté dans ce travail) le plus difficile à résoudre de nos données bio-informatiques (cas : BreastCancer_avecBI) qui était presque impossible à résoudre en utilisant seulement la compression exacte ($930 \rightarrow 231$), devenait presque "facile" avec l'utilisation des compressions majoritaires ($930 \rightarrow 131$). Avec la compression exacte, 2 solutions optimales ont été trouvées après 186 heures de calcul, alors qu'avec la compression majoritaire, l'heuristique SA trouvait quelques dizaines de solutions optimales en moins d'une minute.

Nous avons utilisé deux types de données dans ce travail : les données générées aléatoirement et uniformément et des vraies données de provenance bio-informatique. Nous

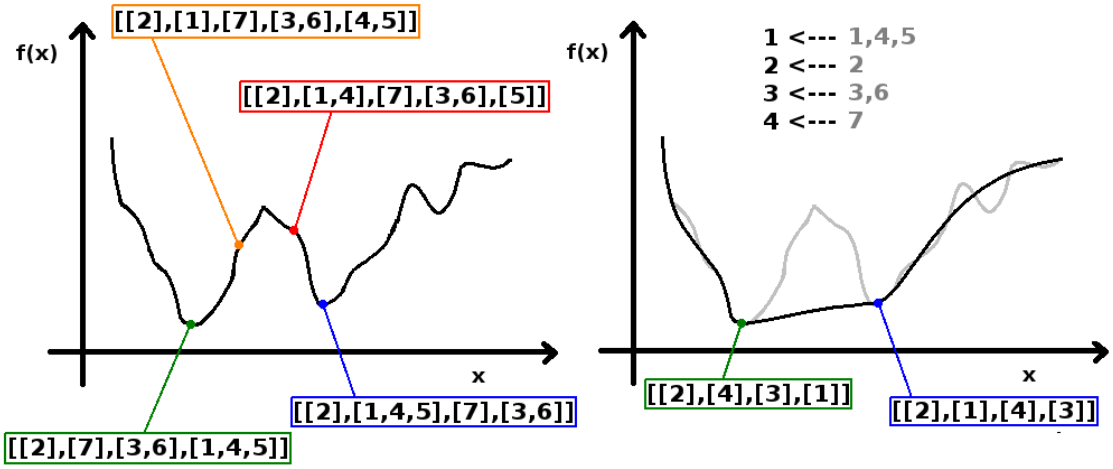


Figure 5.5 – Compression : relaxation du problème. Ici à gauche on illustre la difficulté de passer d'une solution intéressante à une autre solution intéressante dans le cas des classements avec égalité. Alors qu'à droite, on illustre le panorama de l'espace de recherche du même problème, mais après compression. Dans cet exemple, on suppose qu'on a démontré que le groupe des éléments $1,4,5$ est toujours à égalité dans notre ensemble de classements de départ ainsi que le groupe des éléments $3,6$ alors que 2 et 7 sont seuls. La compression permet de naviguer dans un sous-espace de recherche plus pertinent.

avons observé qu'en tirant aléatoirement des classements avec égalité de manière uniforme dans l'espace généralisé $Rank_n$, ces classements présentent très peu d'égalités. De plus, les éléments qui se trouvent dans le même panier dans un classement sont rarement répétés dans d'autres classements. De l'autre côté, les classements de provenance bio-informatique offrent beaucoup d'égalités. L'application des compressions n'a donc pas beaucoup d'impact sur les données aléatoires uniformes, mais fait une différence importante sur les données bio-informatique. Dans nos exemples biologiques, on peut réduire de 28% à 85% la taille des classements (voir tableau 5.II).

5.6 Données réelles et expérimentation

Nous avons utilisé des données bio-informatiques pour paramétrer l'heuristique SA et pour évaluer l'efficacité et la pertinence de notre approche heuristique et de nos algorithmes de compressions. Ces données proviennent de classements de gènes liés à des

maladies. Ces données sont disponibles sur internet[2] et ont été étudiées dans le cadre de la recherche de consensus de classements dans l'article présentant l'heuristique BioConsert [10]. Elles consistent en des classements de gènes par catégories d'importance pour chaque maladie. Un aperçu de la taille et du nombre de ces classements par maladie se trouve au tableau 5.II.

Dans le but de paramétrer l'heuristique SA (voir section 5.4) en fonction du problème, nous avons étudié les paramètres du nombre d'électrons $nbElec$ de départ et le nombre de mouvements $nbMvts$ accordés à chacun de ces électrons. Le paramètre de fission/fusion, pour le choix du mouvement circulaire, a été fixé à $\alpha = 0.6$.

On a remarqué que la probabilité qu'un "électron" (une solution de départ de l'heuristique SA) atteigne une solution optimale dépend de deux facteurs principaux : 1) la difficulté du problème d et 2) le nombre de mouvements $nbMvts$ que l'électron exécute.

On note d la difficulté d'un problème. Elle est définie comme la probabilité potentielle $d \in [0,1]$ pour un électron (au départ de l'heuristique) d'atteindre une solution optimale si on lui laisse la possibilité d'exécuter un grand nombre de mouvements, grand n'étant pas l'infini(< 1000000). La difficulté d d'un problème est faiblement liée à la taille de ses classements de départ (voir Figure I.6 annexe I) ; elle est plutôt liée à la structure inhérente de l'instance. Cela implique que certaines instances sont très difficiles à résoudre alors que d'autres sont plutôt faciles, indépendamment de leur taille. On définit un "électron fructueux" comme étant un électron qui va atteindre une solution optimale après beaucoup de mouvements, d est alors la probabilité qu'un électron soit fructueux. À titre d'exemple, si on lance l'algorithme SA avec 100 éléctrons sur une instance du problème, en accordant un nombre de mouvements par éléctron plus que suffisant, et que parmis ces 100 éléctrons de départ, 48 vont terminer sur une solution optimale, on dira alors que la difficulté du problème est $d = 0.48$. Les autres éléctrons ("non-fructueux") auront vraisemblablement terminè leur trajet dans un minimum local pas optimal. On calcule les statistique sur les éléctrons fructueux.

Pour ce qui est du nombre de mouvements $nbMvts$, il est bien sûr lié à la taille de l'entrée. Nous analysons ici le nombre de mouvements qu'effectue un électron fructueux

[2] http://bioguide-project.net/bioconsert/

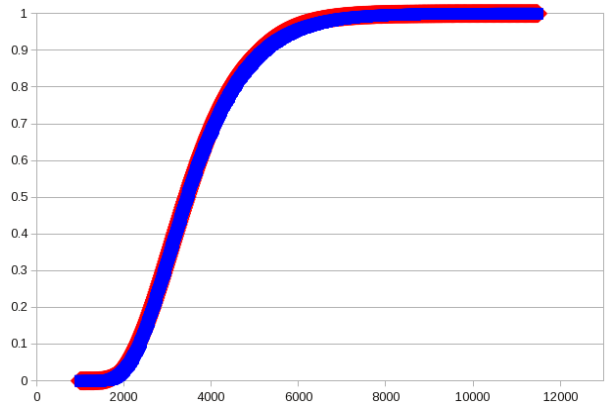


Figure 5.6 – Loi gamma versus le nombre de mouvements nécessaires dans SA. La fonction de distribution bleue est le pourcentage d'électrons fructueux qui ont trouvé une solution optimale après x mouvements. La fonction de distribution rouge est la fonction de distribution de la loi Gamma(5,1) ajustée empiriquement avec un facteur multiplicatif 500 puis un facteur additif 1100 pour obtenir un bon "fit". Pour d'autres exemples d'ajustements de la loi gamma voir la Figure I.5 en annexe I. Les données utilisées pour ce graphique proviennent des classements de gènes liés à la maladie ADHD (voir "cas A" dans le tableau 5.II, ou la Figure I.II en annexe I)

pour atteindre une solution optimale. Un fait observé bien intéressant est que le nombre de mouvements nécessaires pour atteindre une solution optimale (pour un électron fructueux) suit la loi de probabilité Gamma $\Gamma(k,\theta)$ avec paramètres $k = 5$ et $\theta = 1$ avec un facteur multiplicatif *mult* et un facteur additif *add* c.-à-d. $nbMvts \sim add + \Gamma(5,1) \times mult$. Les paramètres *add* et *mult* sont liés à la taille du problème alors que la ressemblance à la loi Gamma semble invariante. Pour chaque cas de maladie, nous avons ajusté les paramètres *add* et *mult* pour avoir la meilleure concordance (le meilleur "fit") de la fonction de distribution de la loi Gamma avec la fonction de distribution des électrons fructueux qui ont atteint une solution optimale après x mouvements. On peut voir cette concordance dans la Figure 5.6.

Après avoir effectués les ajustements de loi Gamma pour toutes les cas de maladies (voir Tableau 5.II), on a noté que les paramètres *add* et *mult* semblent suivre des lois polynomiales en fonction de la taille n_c des instances (compressées exactes) dans le contexte des données bio-informatiques étudiées présentées au début de cette section. On a trouvé : $add = 1.022 n_c^{2.193}$ et $mult = 2.645 n_c^{1.618}$ (voir la Figure 5.7).

On sait maintenant qu'un électron a d comme probabilité d'être fructueux (d'at-

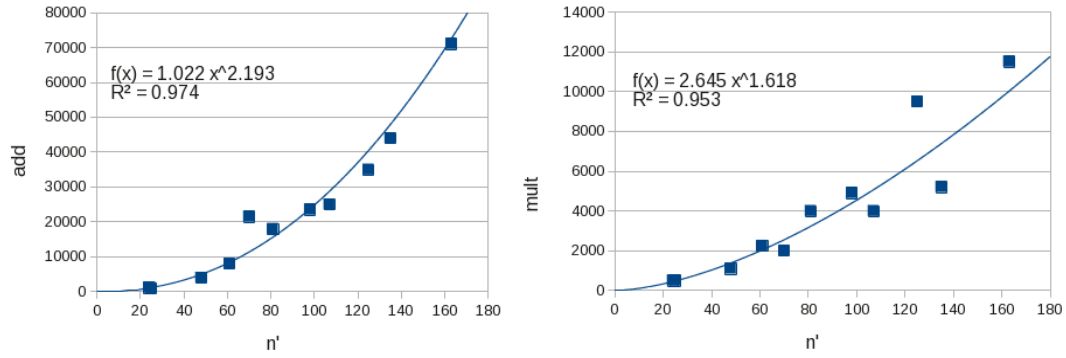


Figure 5.7 – Paramètre *add* et *mult*. Régression simple faite sur les calculs à partir de données de l'application en bio-informatique. Chaque point des graphes est un ajustement de la loi Gamma (Voir Tableau 5.II).

teindre potentiellement une solution optimale), d étant défini par l'instance, et si cet électron est fructueux, le nombre de mouvements $nbMvts$ qu'il va exécuter pour atteindre sa solution optimale va suivre une loi Gamma :

$$nbMvts \sim (1.022 n_c^{2.193}) + \Gamma(5,1) \times (2.645 n_c^{1.618})$$

On est intéressé à paramétrer ce nombre de mouvements en fonction de l'instance pour avoir une certaine mesure d'efficacité. Sachant que le nombre de mouvements suit une loi Gamma, on ne peut pas poser une borne sur $nbMvts$ qui va nous garantir qu'avec ce nombre de mouvements un électron fructueux va toujours atteindre une solution optimale car une réalisation de la loi Gamma peut toujours dépasser une borne arbitraire (son support étant $[0,+\infty[$). On est donc intéressé par une borne réaliste pour une probabilité élevé de succès, disons 90% ou 95%.

Si on désire établir une borne b pour avoir p comme probabilité d'obtenir une réalisation X de la loi Gamma $\Gamma(5,1)$ inférieure à cette borne, il faut regarder simplement l'inverse de la fonction de répartition de la loi Gamma $\Gamma(5,1)$ et on obtient :

$$P(X \leq b) = 0.90, \quad X \sim \Gamma(5,1) \quad \Longrightarrow \quad b = 7.994$$

$$P(X \leq b) = 0.95, \quad X \sim \Gamma(5,1) \quad \Longrightarrow \quad b = 9.155$$

cas	m	n	n_c	$\#Elec$	#succès	%res	add	$mult$	90%	95%
A	4	45	24	200000	80886	0.40443	1100	500	5240	5900
N	4	661	125	24000	2363	0.09846	35000	9500	110400	126400
L	4	35	25	500000	499995	0.99999	1050	500	5075	5725
R	4	402	81	690000	2231	0.00323	18000	4000	50250	54300
P	4	710	163	18000	1167	0.06483	71000	11500	165200	178400
B	4	308	70	720000	10521	0.01461	21500	2000	37950	40050
rP	4	218	61	100000	9378	0.09378	8000	2250	25900	28700
rPs	3	218	48	60000	60000	1.00000	4000	1100	12720	14400
rBC	4	386	98	60000	7528	0.12547	23500	4900	62960	68240
BCs	3	930	135	6700	3932	0.58687	44000	5200	85000	91000
Ps	3	710	107	12000	9673	0.80608	25000	4000	57120	62880

Tableau 5.II – Étude de SA avec cas appliqués. Ici 11 cas sont présentés avec les caractéristiques : m est le nombre de classements, n le nombre d'éléments, n_c le nombre d'éléments après compression exacte. Pour chaque cas, on a lancé $\#Elec$ électrons dans SA (nombre de solutions de départ de l'heuristique), et parmi eux, seulement #succès ont réussi à atteindre une solution optimale, ce qui donne %res de réussite. Pour chaque cas, les meilleurs paramètres add et $mult$ ont été trouvé pour s'ajuster avec la loi Gamma $\Gamma(5,1)$ sur les électrons qui ont été fructueux (voir Figures 5.6 et I.5 en annexe I). Les deux dernières colonnes indiquent le nombre de mouvements nécessaires pour que 90% ou 95% des électrons fructueux atteingnent une solution optimale. Pour chaque cas, l'abréviation A, N, L, etc. représente une maladie et la correspondance entre les abréviations et ces maladies est représenté dans le tableau I.II de l'annexe I.

Avec ces bornes pour la loi Gamma $\Gamma(5,1)$, on peut les insérer dans la formule du nombre de mouvements. On obtient ainsi le nombre de mouvements qu'on doit accorder à un électron, en fonction de la taille n_c compressée du problème, pour qu'il ait 90% ou 95% de probabilité d'atteindre la solution s'il est fructueux. On obtient :

$$nbMvts = (1.022n_c^{2.193}) + 7.994 \times (2.645n_c^{1.618}) \quad pour \;\; 90\%$$

$$nbMvts = (1.022n_c^{2.193}) + 9.155 \times (2.645n_c^{1.618}) \quad pour \;\; 95\%$$

On sait maintenant qu'avec le nombre de mouvements accordés à un électron selon la formule décrite ci-haut, l'électron aura 90% ou 95% comme probabilité de succès d'atteindre une solution optimale s'il est fructueux. Cet électron a aussi $d \in [0,1]$ comme probabilité d'être fructueux. Un électron de départ a alors $0.90 * d$ ou $0.95 * d$ de proba-

bilité d'atteindre une solution optimale avec le nombre de mouvements *nbMvts* suivant les formules ci-haut.

Comme le paradigme du SA le veut, on répète le processus de lancer un électron pour augmenter nos chances de réussite. Si on sait qu'un électron aura $0.95 \times d$ comme probabilité d'atteindre un solution optimale, un nombre $nbElec$ d'électrons nous donnera $1-(1-(0.95 \times d))^{nbElec}$ comme probabilité de trouver une solution optimale avec l'heuristique SA paramétrée. On augmente le nombre d'électrons pour augmenter la probabilité de succès de l'heuristique.

Il reste maintenant à trouver une méthode pour estimer la difficulté d d'un problème basée idéalement sur des caractéristiques observables de l'ensemble de départ.

5.7 Contraintes théoriques - Résumé

Dans cet article, on définit la généralisation du problème de la médiane de permutations au problème de la recherche de médianes pour les classements avec égalités. L'article étudie des propriétés théoriques de ces médianes en généralisant, lorsque possible, des propriétés connues des médianes de permutations.

L'article "Medians of rankings with ties : some theoretical properties" est en réécriture après un refus à une conférence. On va y ajouter d'autres avancées, potentiellement au niveau pratique de la résolution du problème.

5.8 Partage du travail

Dans cet article, j'ai développé et prouvé les théorèmes. Sylvie Hamel et moi avons rédigé l'article.

Medians of rankings with ties: some theoretical properties *

Robin Milosz and Sylvie Hamel

DIRO - Université de Montréal,
QC, Canada

Abstract. Given m permutations and a distance function, the median problem is to find a permutation that is the *closest* of the m given permutations. When considering the Kendall-τ distance that counts the number of pairwise disagreements between permutations, the problem is also known as the Kemeny Score Problem and has been proved to be NP-complete, when $m \geq 4$. Here we investigate the generalized problem of computing a median of a set of m rankings with ties, under a generalized Kendall-τ distance.

1 Introduction

In bioinformatics, we may represent the order of appearance of a common set of n genes in the genomes of m different species by a set of m permutations or signed permutations of $\{1, 2, \ldots, n\}$ (to capture the orientation of the genes in the genomes). In this context, a median permutation, *i.e.* a permutation that is the closest under a given distance d to the m starting permutations, can be viewed as a model for the order of appearance of the n genes in the common ancestor of the m considered genomes. Sankoff and Blanchette [15] have studied this median problem under the breakpoint distance and have derived efficient heuristics to find the median of three circular genomes, with or without the same gene content. Later, the median problem for a set of m permutations or signed permutations under the breakpoint distance was shown to be NP-complete for all $m \geq 3$ by Pe'er and Shamir [14].

The median problem finds applications not only in the context of bioinformatics [14, 15], but also, to name a few, in graph theory [2, 12], applied finance [3], information retrieval and databases [8–11, 13, 16, 17, 20]. In applied finance, the problem of finding the median of m permutations under the Kendall-τ distance (counting the number of order disagreements between pairs of elements of two permutations) is best known under the name "Kemeny Score Problem". In this context, m voters order n politic candidates according to their preferences. The problem then consists in finding a Kemeny consensus, an order of the candidates

* supported by NSERC through an Individual Discovery Grant (Hamel) and by FRQNT through a Master's scholarship (Milosz)

that agrees the most with the orders of the m voters, *i.e.*, that minimizes the sum of the disagreements. This problem was shown to be NP-complete for all $m \geq 4$ [8]. Some approximate and fixed parameter algorithms were later obtained, [2, 3, 12, 20]. With my collaborators, we looked at this median problem for a set of m permutations, under the Kendall-τ distance, for $m \geq 3$, m odd. In [5] we have derived a new efficient heuristic for finding the medians of an odd number of permutations and some nice theoretical properties of these medians.

In real applications, we often do not have strict ordering for our elements *i.e.* we may have ties where some elements are ranked at the same position. The generalized problem of finding a median of ranking with ties is also NP-complete since it contains as a particular case the problem of finding the median of a set of permutations under the Kendall-τ distance. The first heuristic to find a median of rankings with ties was introduced by Fagin and al. in 2004 [9]. Since then, only a few other approaches considered ties [1, 2, 4, 6, 7] leaving several unturned stones for this generalized case. Here, to better understand the problem, we study some theoretical properties of medians of rankings with ties and study their differences with the more known permutation case.

This article is organized as follow. After introducing the general case of the median problem for permutations (in Section 2), we provide, in Section 3, the generalization to rankings with ties along with some necessary definitions and notations. Finally, in sections 4 and 5, we provide our theoretical properties of medians of rankings with ties and some conclusions and futur works.

2 The median problem for permutations

A **permutation** π is a bijection of $[n] = \{1, 2 \ldots, n\}$ onto itself. The set of all permutations of $[n]$ is denoted $\mathcal{S}_n$. As usual, we denote a permutation π of $[n]$ as $\pi = \pi_1 \pi_2 \ldots \pi_n$. The **identity permutation** corresponds to the identity bijection of $[n]$ and is denoted $\imath = 12 \ldots n$. A pair (π_i, π_j) of elements of the permutation π is called an **inversion** if $\pi_i > \pi_j$ and $i < j$. The **number of inversion** of a permutation π is denoted $\text{inv}(\pi)$.

The **Kendall-τ distance**, denoted d_{KT}, counts the number of pairwise disagreements between two permutations and can be defined formally as follows: For permutations π and σ of $[n]$, we have that

$$d_{KT}(\pi, \sigma) = \#\{(i, j) : i < j \textit{ and } [(\pi[i] < \pi[j] \textit{ and } \sigma[i] > \sigma[j]) \textit{ or } (\pi[i] > \pi[j] \textit{ and } \sigma[i] < \sigma[j])\}$$

where $\pi[i]$ denotes the position of integer i in permutation π and $\#\mathcal{S}$ the cardinality of set $\mathcal{S}$. Note that $\text{inv}(\pi) = d_{KT}(\pi, \imath)$.

Given any set of permutations $\mathcal{A} \subseteq \mathcal{S}_n$ and a permutation π, we have $d_{KT}(\pi, \mathcal{A}) = \sum_{\sigma \in \mathcal{A}} d_{KT}(\pi, \sigma)$.

The problem of **finding a median of a set of permutations under the Kendall-τ distance** can be stated formally as follow:

> Given $\mathcal{A} \subseteq \mathcal{S}_n$, we want to find a permutation π^* of $[n]$ such that
>
> $$d_{KT}(\pi^*, \mathcal{A}) \leq d_{KT}(\pi, \mathcal{A}), \text{ for all } \pi \in \mathcal{S}_n.$$

3 Generalizing the problem to ranking with ties

When dealing with rankings with ties, we need a distance that considers two type of disagreements: two elements that are in different orders in the rankings (element i before element j in one ranking and after element j in the other), or two elements that are tied in one ranking and not tied in the other.

Following [9], a **bucket order** on $[n]$ is a transitive binary relation $\lhd$ for which there are non empty sets $\mathcal{B}_1, \ldots, \mathcal{B}_k$ (the **buckets**) that form a partition of $[n]$ such that $x \lhd y$ if and only if there are i, j with $i < j$ such that $x \in \mathcal{B}_i$ and $y \in \mathcal{B}_j$. A **ranking with ties** is then defined on $[n]$ as $R = [\mathcal{B}_1, \ldots \mathcal{B}_k]$, where $R[x] = i$ if $x \in \mathcal{B}_i$. That means that a ranking with ties is simply a surjective function $\sigma : [n] \longrightarrow [k]$, with $\sigma^{-1}(i) = \mathcal{B}_i$.

The **generalized Kendall-τ distance**, denoted $K^{(p)}$, is defined according to a parameter p such that $0 < p \leq 1$:

$$\begin{aligned} K^{(p)}(R_1, R_2) = \#\{(i,j) : i < j \quad & \textit{and } [(R_1[i] < R_1[j] \textit{ and } R_2[i] > R_2[j]), \textit{ or} \\ & (R_1[i] > R_1[j] \textit{ and } R_2[i] < R_2[j]))\} \\ + p \times \#\{(i,j) : i < j \quad & \textit{and } (R_1[i] = R_1[j] \textit{ and } R_2[i] \neq R_2[j]), \textit{ or} \\ & (R_1[i] \neq R_1[j] \textit{ and } R_2[i] = R_2[j])]\} \end{aligned}$$

In this article, we will consider the two kind of disagreements as equally costly, meaning that we will set the parameter p to 1 and will write, for simplicity, $K^{(1)}(R_1, R_2)$ as $K(R_1, R_2)$. Given any set of rankings with ties $\mathcal{R}$ and a ranking r, we have $K(r, \mathcal{R}) = \sum_{R \in \mathcal{R}} K(r, R)$.

The problem of **finding a median of a set of rankings with ties under the generalized Kendall-τ distance** can be stated formally as follows, where $Rank_n$ represents the set of all possible rankings with ties over [n]:

> Given a set of rankings with ties $\mathcal{R} = \{R_1, \ldots R_t\}$, find a ranking with ties R^* of $[n]$ such that
>
> $$K(R^*, \mathcal{R}) \leq K(R, \mathcal{R}), \text{ for all } R \in Rank_n.$$

Since, like stated earlier, a ranking with ties is simply a surjective function $R : [n] \longrightarrow [k]$, we have that if $\mathcal{S}_{n,k}$ represents the number of subjective functions from $[n]$ to $[k]$, then $Rank_n$ contains $\sum_{k=1}^{n} \mathcal{S}_{n,k} = \sum_{k=1}^{n} k! \left\{ {n \atop k} \right\} \sim \frac{n!\ log_2(e)^{n-1}}{2}$

elements, where $\left\{ {n \atop k} \right\}$ represent the Stirling numbers of the second kind and where the asymptotic formula is due to Wilf [19].

Due to its size, searching the whole set of rankings with ties quickly becomes impossible. To be able to compare the results of heuristics for the problem [7, 9] to real medians we need to reduce the search space so that the computation will take place in a reasonable time. This is why, we will study here some theoretical properties of these medians.

4 Theoretical properties of the medians of rankings with ties

When dealing with permutations, we have that given a set $\mathcal{A} \subseteq \mathcal{S}_n$, if a pair of elements appears in the same order in all permutations of $\mathcal{A}$, then they have to appear in that order in a median π^* of $\mathcal{A}$. This was stated and proved in the area of applied finance [18] and uses what they called an Extended Condorcet Criterion. Formally, we have the following theorem.

Theorem 1 *[5, 18] Let $\pi^* = \pi^*_1 \dots \pi^*_n$ be a median of a set of permutations of $[n]$, $\mathcal{A} = \{\pi_1, \pi_2, \dots, \pi_k\} \subseteq \mathcal{S}_n$, under the Kendall-$\tau$ distance. Then if for elements i and j, we have that $\forall \ell$, $1 \leq \ell \leq k$, $\pi_\ell[i] < \pi_\ell[j]$ (resp. $\pi_\ell[i] > \pi_\ell[j]$) then $\pi^*[i] < \pi^*[j]$ (resp. $\pi^*[i] > \pi^*[j]$).* ■

So, in the case of medians of permutations under the Kendall-τ distance, we have that if an element i is always to the left (resp. to the right) of an element j, in all permutations of a set $\mathcal{A}$, then it also has to be to the left (resp. to the right) of j in any median of $\mathcal{A}$.

In the following we will show that the "always" property of Theorem 1 still hold for rankings with ties, meaning that if an elements i is always to the left of (resp. to the right of; tied to) an element j in all rankings of a set of rankings with ties $\mathcal{R}$, then it also has to be to the left of (resp. to the right of; tied to) element j in any median of $\mathcal{R}$.

Theorem 2 *Let $\mathcal{R} = \{R_1, \dots, R_m\}$ be a set of rankings with ties of $[n]$. If two elements i and j, are in the same bucket in all rankings of $\mathcal{R}$, then they will also be in the same bucket in all medians of $\mathcal{R}$. Formally, let R^* be a median of $\mathcal{R} = \{R_1, \dots, R_m\}$ under the generalized Kendall-τ distance. Then if for elements i and j, we have that $\forall \ell$, $1 \leq \ell \leq m$, $R_\ell[i] = R_\ell[j]$ then $R^*[i] = R^*[j]$.*

Proof. Let us define $n \times n$ matrices, L (left), R (right) and E (equal), where L_{ij} (resp. R_{ij} and E_{ij}) denotes the number of rankings with ties in $\mathcal{R}$ that have elements i to the left of element j (resp. to the right of ; equals to). Considering a ranking R, the contribution of a pair of elements i, j to the generalized Kendall-τ distance $K(R, \mathcal{R})$ *i.e.* its disagreements with all rankings of $\mathcal{R}$ is calculated by adding the corresponding cells of the two matrices contradicting the order

between i and j in ranking R. This means that if i is to the left of j in R (resp. to the right of ; equals to), the contribution of this pair of elements to $K(R, \mathcal{R})$ is giving by $R_{ij} + E_{ij}$ (resp. $L_{ij} + E_{ij}$; $R_{ij} + L_{ij}$).

Note that since here elements i and j are always together in all rankings of $\mathcal{R}$ (always in the same bucket), we have that $R_{ij} = 0 = L_{ij}$ and $E_{ij} = m$, which gives us, trivially that,

$$R_{ij} + L_{ij} < E_{ij} + L_{ij} = E_{ij} + R_{ij} \tag{1}$$

It also gives us that for all other element $k \neq i, j$,

$$L_{ik} = L_{jk}, \quad R_{ik} = R_{jk}, \quad E_{ik} = E_{jk}. \tag{2}$$

Now, to prove our theorem, let us assume, by contradiction, that elements i and j are not ties in R^*; without lost of generality, we will consider element i to be to the right of element j and let $R^* = [...[j, l], [k], [i, h]...]$, where l, k and h may represent one element, multiple elements or even none. We will now show that we can lower the generalized Kendall-τ distance of this median R^*, which will contradict our choice of median.

In what follow, we will not change the order of any other pair of elements than those mentioned here (h, i, j, k, l) so we will only compute the contribution of elements h, i, j, k and l to $K(R^*, \mathcal{R})$, considering the rest as a constant. In our computation, we will also always keep the same order between elements h, k and l, so we will consider the cost between those elements to be constant and equal to $\mathcal{C}(l, k, h)$.

$$\begin{aligned} cost([...[j,l],[k],[i,h]...]) = &\overset{(i \text{ right of } j)}{(E_{ij} + L_{ij})} + \overset{(j \text{ tied to } l)}{(L_{jl} + R_{jl}} + \overset{(j \text{ left of } k)}{E_{jk}} + R_{jk} + \overset{(j \text{ left of } h)}{E_{jh}} + R_{jh}) \\ &+ (\overset{(i \text{ right of } l)}{L_{il}} + E_{il} + \overset{(i \text{ right of } k)}{L_{ik}} + E_{ik} + \overset{(i \text{ tied to } h)}{L_{ih}} + R_{ih}) + \mathcal{C}(l,k,h) \end{aligned}$$

If $\overset{(\text{disagreements of } j \text{ with } h,k \text{ and } l)}{(L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh})} \leq \overset{(\text{disagreements of } i \text{ with } h,k \text{ and } l)}{(L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih})}$, equation (2) give us that

$$(L_{il} + R_{il} + E_{ik} + R_{ik} + E_{ih} + R_{ih}) \leq (L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih}) \tag{3}$$

which gives us

$$\begin{aligned} cost([...[j,l],[k],[i,h]...]) &\overset{(3)}{\geq} (E_{ij} + L_{ij}) + (L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh}) \\ &\quad + (L_{il} + R_{il} + E_{ik} + R_{ik} + E_{ih} + R_{ih}) + \mathcal{C}(l,k,h) \\ &\overset{(1)}{>} (R_{ij} + L_{ij}) + (L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh}) \\ &\quad + (L_{il} + R_{il} + E_{ik} + R_{ik} + E_{ih} + R_{ih}) + \mathcal{C}(l,k,h) \\ &= cost([...[i,j,l],[k],[h]...]) \end{aligned}$$

Now, if we rather have

$$\overbrace{(L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih})}^{\text{(disagreements of } i \text{ with } h,k \text{ and } l)} \leq \overbrace{(L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh})}^{\text{(disagreements of } i \text{ with } h,k \text{ and } l)},$$

equation (2) give us that

$$(L_{jl} + E_{jl} + L_{jk} + E_{jk} + L_{jh} + R_{jh}) \leq (L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh}) \quad (4)$$

and then

$$\begin{aligned} cost([...[j,l],[k],[i,h]...]) &\overset{(4)}{\geq} (E_{ij} + L_{ij}) + (L_{jl} + E_{jl} + L_{jk} + E_{jk} + L_{jh} + R_{jh}) \\ &\quad + (L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih}) + \mathcal{C}(l,k,h) \\ &\overset{(1)}{>} (R_{ij} + L_{ij}) + (L_{jl} + E_{jl} + L_{jk} + E_{jk} + L_{jh} + R_{jh}) \\ &\quad + (L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih}) + \mathcal{C}(l,k,h) \\ &= cost([...[l],[k],[i,j,h]...]) \end{aligned}$$

This show that a ranking with ties that does not have i and j in the same bucket in not optimal, leading to a contradiction. Like stated earlier, elements k, h and l in this demonstration may represent one element, multiple elements or even none. In the case of none, the values implying k, h or l are all set to 0. In the case of multiple elements, the demonstration is still valid because the equations are valid for each represented element. ■

Theorem 3 *Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a set of rankings with ties of $[n]$. If i is to the left of j (resp. to the right of j) in all rankings of $\mathcal{R}$, then it will also be to the left of j (resp. to the right of j) in all medians of $\mathcal{R}$. Formally, let R^* be a median of $\mathcal{R} = \{R_1, \ldots, R_m\}$ under the generalized Kendall-τ distance. Then if for elements i and j, we have that $\forall \ell$, $1 \leq \ell \leq m$, $R_\ell[i] < R_\ell[j]$, (resp. $R_\ell[i] > R_\ell[j]$) then $R^*[i] < R^*[j]$ (resp. $R^*[i] > R^*[j]$).*

Proof. We will prove the case where i is the the left of j in all rankings of $\mathcal{R}$, the case where i is to the right of j in all rankings of $\mathcal{R}$ will follow by symmetry.

Like in the proof of Theorem 2 let us define $n \times n$ matrices, L (left), R (right) and E (equal), where L_{ij} (resp. R_{ij} and E_{ij}) denotes the number of rankings with ties in $\mathcal{R}$ that have elements i to the left of element j (resp. to the right of and equals to).

By contradiction, we will investigate the two following case: 1) R^* is a median of $\mathcal{R}$ and element i is tied to element j (they are in the same bucket), 2) R^* is a median of $\mathcal{R}$ and element i is to the right of element j.

1) R^* is a median of $\mathcal{R}$ and element i is tied to element j:
Let $R^* = [...[i,j,k]...]$, where k may represent one element, multiple elements or even none. First we can inspect all the possible positions, in the rankings with ties of $\mathcal{R}$, of this $k \neq i, j$.

Table 1. Different possible positions of element k in rankings with ties of $\mathcal{R}$

k	i	...	j	...
...	[i,k]	...	j	...
...	i	k	j	...
...	i	...	[j,k]	...
...	i	...	j	k

We can derive two useful inequalities from this Table 1. For any element $k \neq i, j$, the number of rankings with ties in $\mathcal{R}$ where j is to the right of k is at least the number of rankings with ties where i is to the right of k or tied with k. This is simply because i is always to the left of j. Symmetrically, we can say the same for the number of rankings with ties where i is on the left of k. It is at least the number of times that j is on the left side of k or tied with it. Giving us:

$$R_{jk} \geq R_{ik} + E_{ik} \tag{5}$$

$$L_{ik} \geq L_{jk} + E_{jk} \tag{6}$$

Since i is to the left of j in all rankings of $\mathcal{R}$ we also have that $R_{ij} = E_{ij} = 0$ and $L_{ij} = m$, which gives us trivially that

$$R_{ij} + E_{ij} < R_{ij} + L_{ij} \tag{7}$$

Finally, we show that we can lower the generalized Kendall-τ distance of our median R^* in which elements i, j and k are in the same bucket by applying two transpositions, which will contradict our choice of median and rise a contradiction. In what follow, we will not change the order of any other pair of elements than those mentioned here (i, j, k) so we will only compute the contribution of elements i, j and k in $K(R^*, \mathcal{R})$, considering the rest as a constant. For clarity, the changes at each steps have been underlined.

$$\begin{aligned}
cost([...[i,j,k]...]) &= \underline{R_{ij} + L_{ij}} + R_{ik} + L_{ik} + R_{jk} + L_{jk} \\
&\overset{(7)}{>} R_{ij} + E_{ij} + R_{ik} + \underline{L_{ik}} + \underline{R_{jk}} + L_{jk} \\
&\overset{(5),(6)}{\geq} R_{ij} + E_{ij} + R_{ik} + L_{jk} + E_{jk} + R_{ik} + E_{ik} + L_{jk} \\
&= R_{ij} + E_{ij} + \underline{2R_{ik}} + E_{ik} + \underline{2L_{jk}} + E_{jk} \\
&\geq \overset{(i \text{ left of } j)}{(R_{ij} + E_{ij})} + \overset{(i \text{ left of } k)}{(R_{ik} + E_{ik})} + \overset{(j \text{ right of } k)}{(L_{jk} + E_{jk})} \\
&= cost([...[i],[k],[j]...])
\end{aligned}$$

This show that a ranking with ties that have i and j in the same bucket does not have a minimal score. The element k in this demonstration may represent one

element, multiple elements or even none, like in the proof of Theorem 2. Having supposed a median permutation that has i and j in the same bucket leads to a contradiction.

2) R^* is a median of $\mathcal{R}$ and element i is to the right of element j:
Let $R^* = [...[j,l],[k],[i,h]...]$, where l, k and h may represent one element, multiple elements or even none. First we can inspect all the possible positions, in the rankings with ties of $\mathcal{R}$, of elements h, k and l. (See Table 1, where k can be replaced by h or l). From those tables, we can derived, like we did in preceding case, these inequalities, where ℓ is either h, k or l:

$$R_{j\ell} \geq R_{i\ell} + E_{i\ell} \tag{8}$$

$$L_{i\ell} \geq L_{j\ell} + E_{j\ell} \tag{9}$$

Following the same steps, since i is left to j in all rankings of $\mathcal{R}$ we have again that

$$R_{ij} + E_{ij} < R_{ij} + L_{ij} \tag{10}$$

Finally, we show that we can lower the generalized Kendall-τ distance of our median R^* in which elements element i is to the right of element j, which will contradict our choice of median. In what follow, we will not change the order of any other pair of elements than those mentioned here (h, i, j, k, l) so we will only compute the contribution of elements h, i, j, k and l in $K(R^*, \mathcal{R})$, considering the rest as a constant. We will also keep the same order between elements h, k and l so we consider the cost between those elements to be also constant and equal to $\mathcal{C}(l, k, h)$. For clarity, the changes at each steps have been underlined.

$$\begin{aligned}
cost([...[j,l],[k],[i,h]...]) &= \underline{E_{ij} + L_{ij}} + L_{jl} + R_{jl} + E_{jk} + R_{jk} + E_{jh} + R_{jh} \\
&\quad + L_{il} + E_{il} + L_{ik} + E_{ik} + L_{ih} + R_{ih} + \mathcal{C}(l,k,h) \\
&\overset{(10)}{>} E_{ij} + R_{ij} + L_{jl} + \underline{R_{jl}} + E_{jk} + \underline{R_{jk}} + E_{jh} + \underline{R_{jh}} \\
&\quad + \underline{L_{il}} + E_{il} + \underline{L_{ik}} + E_{ik} + \underline{L_{ih}} + R_{ih} + \mathcal{C}(l,k,h) \\
&\overset{(8),(9)}{\geq} E_{ij} + R_{ij} + L_{jl} + R_{il} + E_{il} + E_{jk} + R_{ik} + E_{ik} \\
&\quad + E_{jh} + R_{ih} + E_{ih} + L_{jl} + E_{jl} + E_{il} + L_{jk} + \\
&\quad E_{jk} + E_{ik} + L_{jh} + E_{jh} + R_{ih} + \mathcal{C}(l,k,h) \\
&= E_{ij} + R_{ij} + \underline{2L_{jl}} + R_{il} + \underline{2E_{il} + 2E_{jk}} + R_{ik} + \\
&\quad \underline{2E_{ik} + 2E_{jh} + 2R_{ih}} + E_{ih} + E_{jl} + L_{jk} + L_{jh} \\
&\quad + \mathcal{C}(l,k,h) \\
&\geq (E_{ij} + R_{ij} + R_{ih} + E_{ih} + R_{ik} + E_{ik} + R_{il} + E_{il}) \\
&\quad \overset{(i \text{ left of } j,h,k \text{ and } l)}{} \\
&\quad + (L_{jh} + E_{jh} + L_{jk} + E_{jk} + L_{jl} + E_{jl}) + \mathcal{C}(l,k,h) \\
&\quad \overset{(j \text{ right of } h,k \text{ and } l)}{} \\
&= cost([...[i],[l],[k],[h],[j]...])
\end{aligned}$$

This show that a ranking with ties that have element i to the right of element j does not have a minimal score. The element k, l and h in this demonstration may represent one element, multiple elements or even none, like before. Having supposed a median ranking that has i to the right j leads to a contradiction. ■

5 Conclusion and Future works

We have showed that, when working with a generalized Kendall-τ distance, if a pair of elements is always in the same order in all rankings of a set than it has to be in that order in any median of this set, generalizing a result already known for permutations. An interesting fact is that, when dealing with permutations, *i.e.* with a strict order, this "always in the same order" implies automatically its "never" conterpart, that is if an element i is never to the left (resp. to the right) of an element j, in all permutations of a set $\mathcal{A}$, then it cannot be to the left (resp. to the right) of j in any median of $\mathcal{A}$.

We have computed with an exact algorithm the following examples that show that this "never" parts of Theorem 1 do not hold for rankings with ties:

Example 1 : Never tied in $\mathcal{R}$ $\not\Rightarrow$ not tied in a median R^*
Let $\mathcal{R} = \{[[i, X], [j]], [[i], [j, X]], [[j, X], [i]], [[j], [i, X]]\}$, where X is a set of elements, where $|X| \geq 4$. Even if elements i and j are never in the same bucket in the four rankings with ties of $\mathcal{R}$, a median for this set is $R^* = [[i, j, X]]$, which ties i and j.

Example 2 : Never right/left in $\mathcal{R}$ $\not\Rightarrow$ never right/left in a median R^*
Let $\mathcal{R} = \{[[Y], [i, X], [j]], [[i], [j, Y], [X]], [[i, j, X, Y]], [[i, j, Y], [X]], [[Y], [i, X], [j]]\}$, where X and Y are set of elements, where $|X| \geq 5$ and $|Y| \geq 5$. Even if element i is never to the right of element j in the five rankings with ties of $\mathcal{R}$, a median for this set is $R^* = [[j, Y], [i, X]]$, where i is right of element j.

As futur works, we would like to continue to investigate medians of rankings with ties to access their differences and similarities with the permutation case. Another interesting path would be to investigate some set properties of the medians; given two different starting sets of rankings with ties $\mathcal{R}_1$ and $\mathcal{R}_2$ that have, respectively, as set of medians $\mathcal{M}(\mathcal{R}_1)$ and $\mathcal{M}(\mathcal{R}_2)$, what can we say about the sets of medians $\mathcal{M}(\mathcal{R}_1 \cup \mathcal{R}_2)$, $\mathcal{M}(\mathcal{R}_1 \cap \mathcal{R}_2)$, $\mathcal{M}(\mathcal{R}_1 \setminus \mathcal{R}_2)$, $\mathcal{M}(\overline{\mathcal{R}_1})$? What about the special sets $\mathcal{R}$ with $\mathcal{M}(\mathcal{R}) \subseteq \mathcal{R}$ or $\mathcal{M}(\mathcal{R}) \cap \mathcal{R} = \emptyset$?

Also, following [4], we would like to see if we can derive efficient exact algorithms for the median of rankings with ties problem when the rankings considered show a sufficiently high degree of similarity between each other on average.

References

1. N. Ailon, *Aggregation of partiel rankings, p-ratings and top-m lists*, Algorithmica, 57(2), pp. 284–300, 2010.
2. N. Ailon, M. Charikar and N. Newman, *Aggregating inconsistent information: Ranking and clustering*, Journal of the ACM, 55(5), article 23, 2008.
3. N. Betzler, M.R. Fellows, J. Guo, R. Niedermeier and F.A. Rosamond, *Fixed-Parameter Algorithms for Kemeny Scores*, LNCS 5034, pp. 60–71, 2008.
4. N. Betzler, J. Guo, C. Komusiewicz and R. Niedermeier, *Average Parameterization and Partial Kernelization for Computing Medians*, J. Comput. Syst. Sci, 77-4, pp. 774–789, 2011.
5. G. Blin, M. Crochemore, S. Hamel et S. Vialette, *Medians of an odd number of permutations*, Pure Mathematics and Applications, Volume 21, Issue 2, pp. 161–175, 2010.
6. B. Brancotte, B. Rance, A. Denise and S. Cohen-Boulakia, *Conqur-bio: Consensus rankings with query reformulation for biological data*, In Data Integration in the Life Sciences, LNCS 8574, pp. 128–142, 2014.
7. S. Cohen-Boulakia, A. Denise and S. Hamel, *Using medians to generate consensus rankings for biological data*, In Scientific and Statistical Datebase Management, LNCS 6809, pp. 73–90, 2011.
8. C. Dwork, R. Kumar, M. naor and D. Sivakumar, *Rank Aggregation Methods for the Web*, in proceedings of the 10th WWW, pp.613-622, 2001.
9. R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, *Comparing and aggregating rankings with ties*, In Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp.47-58, 2004.
10. R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, *Comparing partial rankings*, SIAM Journal on Discrete Mathematics, 20(3), pp. 628–648, 2006.
11. R. Fagin, R. Kumar, and D. Sivakumar. *Comparing top k lists*, SIAM Journal on Discrete Mathematics,17(1), pp.134–160, 2003.
12. C. Kenyon-Mathieu and W. Schudy, *How to rank with few errors*, In Proceedings of the 39th STOC, pp. 95-103, 2007.
13. M. Jacob, B. Kimelfeld, and J. Stoyanovich, *A system for management and analysis of preference data*, Proc. of the VLDB Endowment, 7(12), 2014.
14. Pe'er, I and Shamir, R., *The median problems for breakpoints are NP-complete*, Elec. Colloq. on Comput. Complexity, vol. 71, 1998.
15. Sankoff, D. and Blanchette, M., *The median problem for breakpoints in comparative genomics*, LNCS 1276, pages 251–263, 1997.
16. F. Schalekamp and A. van Zuylen. *Rank aggregation: Together we're strong*, In ALENEX, pp. 38–51, SIAM, 2009.
17. J. Stoyanovich, S. Amer-Yahia, S. B. Davidson, M. Jacob, T. Milo, et al., *Understanding local structure in ranked datasets*, In CIDR, 2013.
18. M. Truchon, *An Extension of the Condorcet Criterion and Kemeny Orders*, Internal Report, cahier 98-15 du Centre de Recherche en Économie et Finance Appliquées, Université Laval, 16 pages, 1998.
19. H. S. Wilf, *Generatingfunctionology*, Academic Press, NY, p. 147, 1990.
20. A. vanZuylen and D.P. Williamson, *Deterministic algorithms for rank aggregation and other ranking and clustering problems*, in Proceedings of the 5th WAOA, LNCS 4927, pp.260–273, 2007.

CONCLUSION

Résumé

Le problème de la médiane de permutations est un problème combinatoire qui consiste à trouver une permutation qui est la plus proche d'un ensemble donné de permutations. On utilise la distance de Kendall-τ qui compte le nombre de paires d'éléments dont l'ordre est inversé. Le problème de la médiane de permutation est un problème NP-Difficile.

Dans le travail de ce mémoire, nous avons introduit formellement le problème de la médiane de permutations et son contexte d'application. Nous avons approché le problème sous trois angles différents pour sa résolution et nous avons introduit une généralisation aux classements avec égalités.

On présente des méthodes pour résoudre le problème soit de façon exacte (algorithme Branch and Bound) pour des instances de petite taille ($n < 25$) soit une bonne approximation (heuristique Simulated Annealing) pour des instances de plus grande taille ($n < 200$).

Des contraintes ont été proposées qui permettent de réduire l'espace de recherche d'une manière non négligeable. Ce prétraitement fixe l'ordre de certaines paires d'éléments en se basant sur l'interférence entre les éléments de cette paire dans l'ensemble de départ. Ces contraintes peuvent ordonner jusqu'en moyenne 35% des paires d'éléments pour des instances aléatoires de $m = 3$ et $n = 100$ ou 34% pour $m = 50$ et $n = 30$ et beaucoup plus avec des m et n petits, où m est le nombre de permutations dans l'ensemble de départ et n leur taille.

Si le problème est en général considéré difficile, ce n'est pas nécessairement le cas de toutes les classes d'instances. Une classe particulière d'instances du problème a été montrée facile à résoudre. Cette classe est une sous-classe de la classe des invariants à l'opération médiane : $A = M(A)$.

Finalement, poussée par des applications réelles, une généralisation du problème a été proposée suivie de l'adaptation de l'algorithme Branch and Bound et de l'heuristique

du Simulated Annealing de la section 2 à cette généralisation. Une compression des données nous a permis d'accélérer les calculs et de rendre le problème plus facile à résoudre. Le prétraitement des données par compression et l'heuristique de résolution SA ont été appliqué à des données de provenance biologique pour évaluer leur performance. Une étude sur la généralisation de propriétés théoriques des médianes de permutations aux classements avec égalités a montré que certaines de ces propriétés restent vraies alors que d'autres tombent.

Travaux futurs

Avec ce travail plusieurs voies s'ouvrent pour des travaux futurs tant au niveau pratique qu'au niveau théorique.

D'un point de vue plus pratique et algorithmique, le raffinement de la borne inférieure d'élagage ainsi que l'étude de techniques pour éviter le recalcul de sous-cas indentiques nous permettraient d'eméliorer grandement l'efficacité de notre Branch and Bound exact.

Pour ce qui est de notre approche heuristique (Simulated Annealing), elle ne gère efficacement que des instances de taille 200 ou moins. Plusieurs avenues sont intéressantes à investiguer dans le but de pouvoir résoudre des problèmes de très grande taille (big data). De nouvelles compressions de données peuvent être étudiées de même qu'une optimisation locale par déplacement d'une fenêtre de taille k.

D'un point de vue théorique, l'étude des propriétés combinatoires des médianes de permutations et de classements avec égalité continuera d'apporter l'information nécessaire aux développements de nouvelles contraintes permettant de diriger la recherche de médianes.

Le cas $m = 3$ est particulièrement intéressant à investiguer d'un point de vue théorique étant donné que la compléxité du problème de la médiane de trois permutations est encore inconnue. De plus, ce cas est souvent beaucoup plus rapide à calculer que les autres tailles d'ensemble m plus grandes pour des tailles de permutations n identiques. Il serait alors intéressant de savoir s'il existe un algorithme polynomial pour résoudre ce

cas ou, dans le cas contraire, trouver une heuristique qui y est bien adaptée.

L'ensemble des permutations médianes $M(A)$ présente souvent des motifs dans ses permutations avec des régions plus variables et d'autres qui paraissent plus stables. Décomposer le problème pour y trouver des structures qui peuvent donner des indices sur la forme finale des médianes serait pertinent dans une optique d'améliorer les approches algorithmiques.

Finalement, complètement caractériser le cas des ensembles automédians $A = M(A)$ est un autre problème théorique intéressant, de même qu'investiguer d'autres sous-classes de permutations, comme les classes d'équivalence de Knuth [24], pour lesquelles le problème de la médiane est possiblement polynomial.

BIBLIOGRAPHIE

[1] N. Ailon. Aggregation of partiel rankings, p-ratings and top-m lists. *Algorithmica*, 57(2), 2010.

[2] N. Ailon, M. Charikar et N. Newman. Aggregating inconsistent information : ranking and clustering. *Journal of the ACM*, 55(5):pp 1–27, 2008.

[3] A. Ali et M. Meila. Experiments with kemeny ranking : What works when ? *Mathematical Social Sciences*, 64, 2012.

[4] A. Denise B. Brancotte, B. Rance et S. Cohen-Boulakia. Conqur-bio : Consensus rankings with query reformulation for biological data. *Data Integration in the Life Sciences*, LNCS 8574, 2014.

[5] N. Betzler et al. Average parameterization and partial kernelization for computing medians. *Journal of Computer and System Sciences*, 77(4):774–789, 2011.

[6] N. Betzler, R. Bredereck et R. Niedermeier. Theoretical and empirical evaluation of data reduction for exact kemeny rank aggregation. *Autonomous Agents and Multi-Agent Systems*, 28(1):721–748, 2014.

[7] T. Biedl, F.J. Brandenburg et X. Deng. Crossings and permutations. *LNCS*, 3843: 1–12, 2005.

[8] Guillaume Blin, Maxime Crochemore, Sylvie Hamel et Stéphane Vialette. Median of an odd number of permutations. *Pure Mathematics and Applications*, 21(2), 2011.

[9] R. ; Cohen, W. ; Schapire et Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10, 1999.

[10] Sarah Cohen-Boulakia, Alain Denise et Sylvie Hamel. Using medians to generate consensus rankings for biological data. *Lecture Notes in Computer Science*, 6809 (1), 2011.

[11] Davenport A. Kalagnanam J. Conitzer, V. Improved bounds for computing kemeny rankings. Dans *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 620–627, 2006.

[12] Benjamin Correal et Philippe Galinier. On the complexity of searching the linear ordering problem neighborhoods. *Evolutionary Computation in Combinatorial Optimization*, 9026(1), 2011.

[13] Kalagnanam J. Davenport, A. A computational study of the kemeny rule for preference aggregation. Dans *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 697–702, 2004.

[14] R. P. DeConde, S. Hawley, S. Falcon, N. Clegg, B. Knudsen, et R. Etzioni. Combining results of microarray experiments : a rank aggregation approach. *Statistical Applications in Genetics and Molecular Biology*, 5(1), 2006.

[15] P. Diaconis et R. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society, Series B*, 39(2), 1977.

[16] C. Dwork, R. Kumar, M. Naor et D. Sivakumar. Rank aggregation methods for the web. *in proceedings of the 10th WWW*, pages 613–622, 2001.

[17] R.A. Fisher et F. Yates. Statistical tables for biological, agricultural and medical research, london : Oliver and boyd. 3:26–27, 1948.

[18] Sylvie Hamel et Robin Milosz. Medians of permutations : when A = M(A). Dans *Procceedings of 12th International Permutation Patterns Conference*, pages 68–71, 2014.

[19] M. Karpinski et W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. *LNCS*, 6506:3–14, 2010.

[20] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:577–591, 1959.

[21] M. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–89, 1938.

[22] C. Kenyon-Mathieu et W. Schudy. How to rank with few errors. *STOC07*, pages 95–103, 2007.

[23] S. Kirkpatrick, C. D. Gelatt Jr et M. P Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[24] D. Knuth. Permutations matrices and generalized young tableaux. *Pacif J. Math*, 34, 1970.

[25] D.E. Knuth. Seminumerical algorithms. *The Art of Computer Programming 2 Reading, MA : AddisonWesley*, pages 124–125, 1969.

[26] P.J.M. Van Laarhoven et E.H.L. Aarts. *Simulated annealing : theory and applications*. Kluwer Academic Publishers, 1987.

[27] A. H. Land et A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3), 1960.

[28] Phadnis K. Patterson A. Bilmes J. Meila, M. Consensus ranking under the exponential model. Dans *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence*, pages 285–294, 2007.

[29] Robin Milosz et Sylvie Hamel. Medians of permutations : building constraints. Dans *Proceedings of 2nd Conference on Algorithms and Discrete Applied Mathematics*. accepté, 2016.

[30] J. Guo R. Niedermeier N. Betzler, M. R. Fellows et F. A. Rosa-mond. Fixed-parameter algorithms for kemeny scores. Dans *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, pages 60–71, 2008.

[31] J. Guo R. Niedermeier N. Betzler, M. R. Fellows et F. A. Rosamond. How similarity helps to efficiently compute kemeny rankings. Dans *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[32] N. Nishimura et N. Simjour. Parameterized enumeration of (locally-) optimal aggregations. *IWADS13, LNCS*, 8037:512–523, 2013.

[33] M. Mahdian D. Sivakumar R. Fagin, R. Kumar et E. Vee. Comparing and aggregating rankings with ties. Dans *In Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 47–58, 2004.

[34] J. Sese et S. Morishita. Rank aggregation method for biological databases. *Genome informatic series*, pages 506–507, 2001.

[35] N. Simjour. Improved parameterized algorithms for the kemeny aggregation problem. *IWPEC09, LNCS*, 5917:312–323, 2009.

[36] J. Starlinger, B. Brancotte, S. Cohen-Boulakia et U. Leser. Similarity search for scientific workflows. *Proceedings of the VLDB Endowment*, 7(12), 2014.

[37] M. Truchon. An extension of the condorcet criterion and kemeny orders. *Internal Report, Université Laval*, page 16 pages, 1998.

[38] A. van Zuylen et D.P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research*, 34(3): 594–620, 2009.

[39] V. Černý. Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1), 1985.

Annexe I

Figures, graphiques et tableaux complémentaires au mémoire

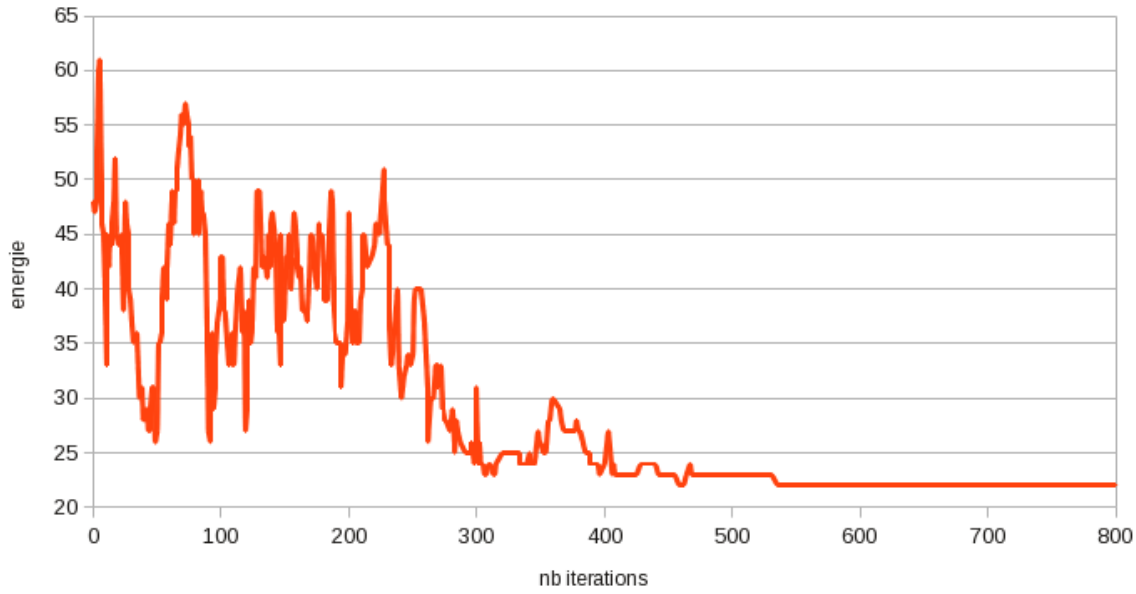


Figure I.1 – SA : graphique de l'énergie de la solution courante. Exemple en application. On remarque un haute variabilité au début quand la température est haute, suivi d'un refroidissement qui fait converger la solution courante puis un état stable à la fin. C'est l'application de l'heuristique SA présentée dans le pseudo-code Algortithm 2 de la page 15 qui a été faite sur l'exemple de la Figure 1.1 (modélisé en permutations à la Figure 2.6). Les Figures suivantes I.2 et I.3 sont d'autres exemples d'exécutions.

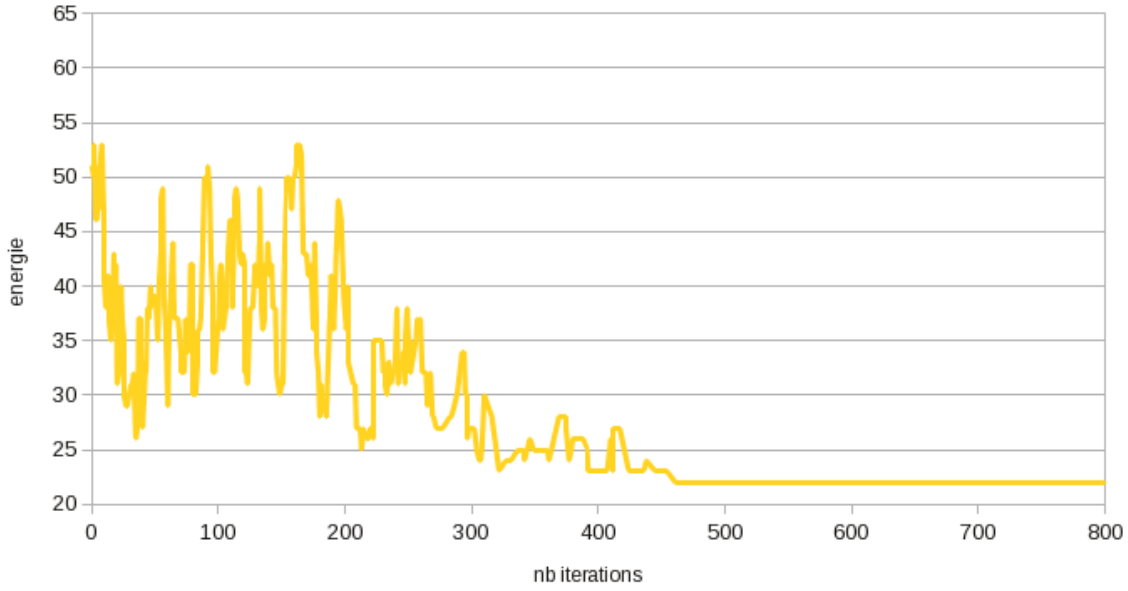


Figure I.2 – SA : graphique de l'énergie de la solution courante. Exemple en application. Les détails de la génération de ce grpahiques sont donnés à la Figure I.1.

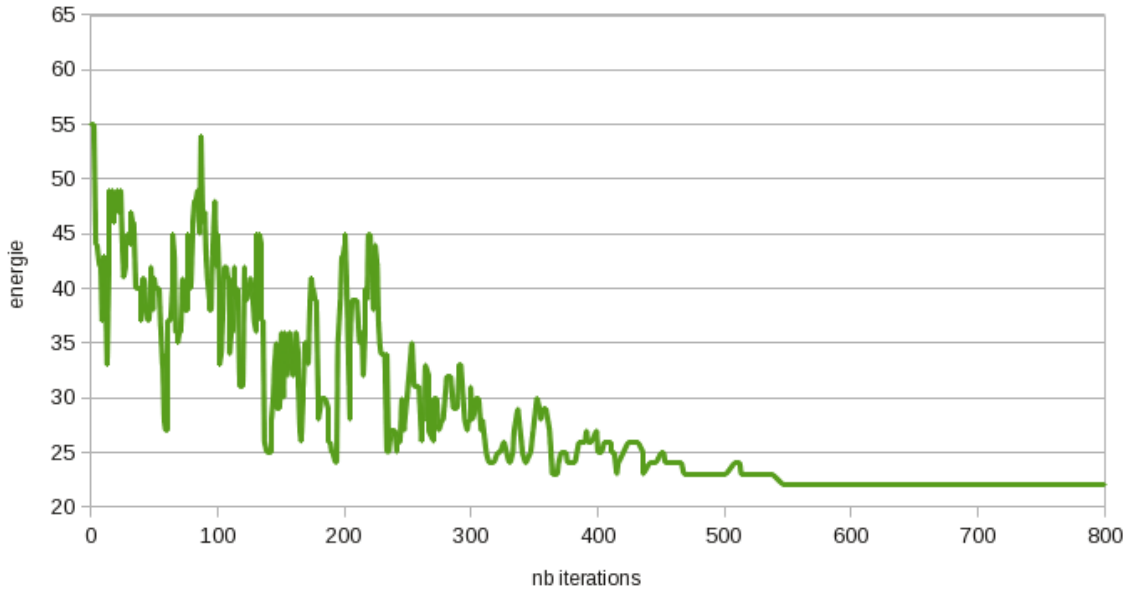


Figure I.3 – SA : graphique de l'énergie de la solution courante. Exemple en application. Les détails de la génération de ce grpahiques sont donnés à la Figure I.1

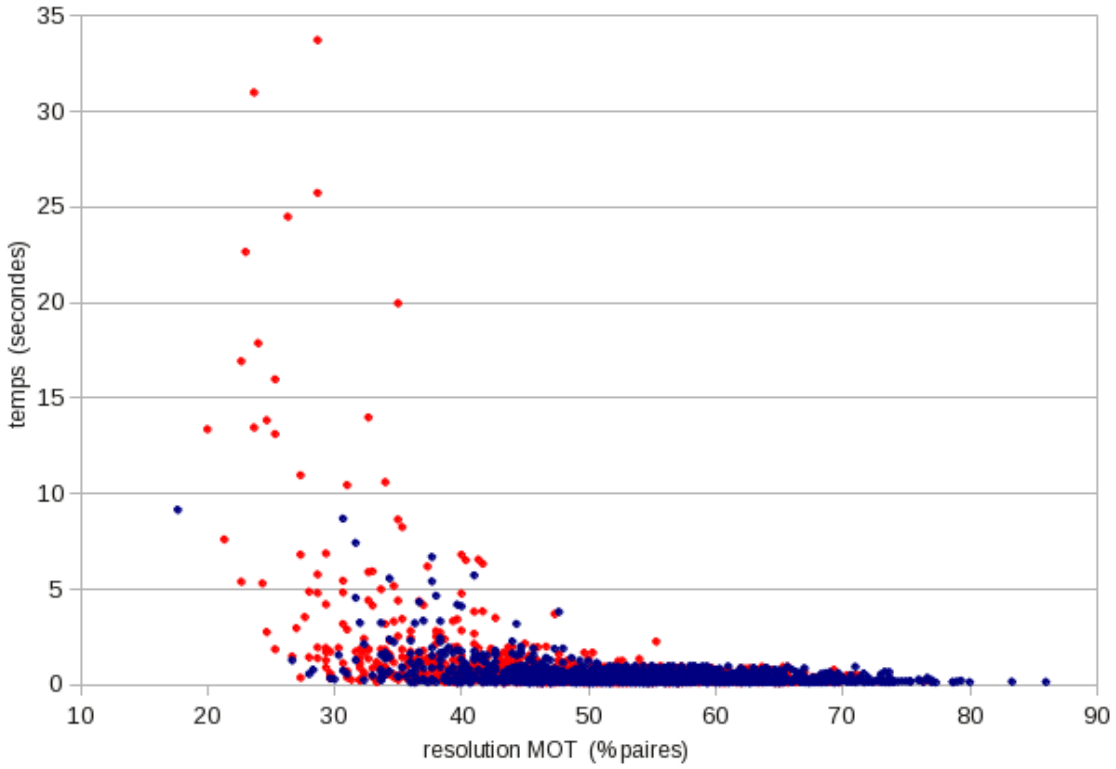


Figure I.4 – Temps d'exécution du B&B par rapport à la résolution par contraintes MOT. Statistique basée sur 1000 simulations de $m = 3$ et $n = 25$ (en bleu) puis 1000 simulations de $m = 5$ et $n = 25$ (en rouge).

$m \backslash n$	8	10	12	14	15	20	25	30
3	1000	1000	1000	500	1000	1000	1000	100
4	1000	1000	1000	500	500	-	-	-
5	1000	1000	1000	500	1000	1000	1000	100
6	1000	1000	1000	500	500	-	-	-
10	1000	1000	1000	500	1000	500	-	-
15	1000	1000	1000	1000	1000	1000	500	-
20	1000	1000	1000	1000	1000	1000	-	-
25	1000	1000	1000	1000	1000	1000	500	-

Tableau I.I – Tableau du nombre d'instances aléatoires simulées pour faire les statistiques pour les tableaux 2.I, 2.II et 2.III des sections 2.3.1 et 2.4. (- dénote une donnée non disponible)

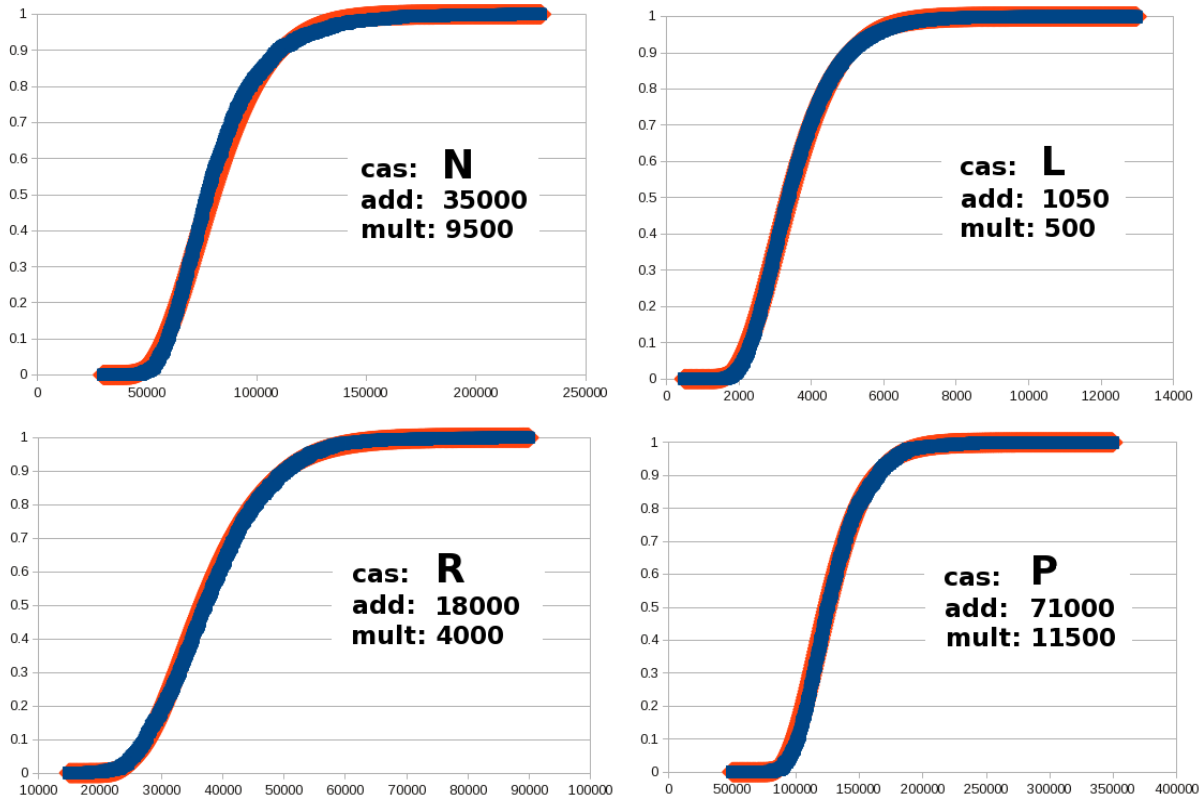


Figure I.5 – Loi gamma vs le nombre de mouvements nécessaires dans SA. La fonction de distribution bleue est le pourcentage d'électrons qui ont trouvé une solution optimale après x mouvements. La fonction de distribution rouge est la fonction de distribution de la loi Gamma(5,1) ajustée empiriquement avec un facteur additif et un facteur multiplicatif pour obtenir un bon "fit". (cas présentés : L : LongQtSyndrome, N : Neuroblastoma, R : Retinoblastoma et P : Prostate-Cancer)

abréviation	cas
A	ADHD_avecBI
N	Neuroblastoma_avecBI
L	LongQtSyndrome_avecBI
R	Retinoblastoma_avecBI
P	ProstateCancer_avecBI
B	BladderCancer_avecBI
rP	ReducedProstateCancer_avecBI
rPs	ReducedProstateCancer_sansBI
rBC	ReducedBreastCancer_avecBI
BCs	BreastCancer_sansBI
Ps	ProstateCancer_sansBI

Tableau I.II – Noms complets des cas appliqués en bio-informatique

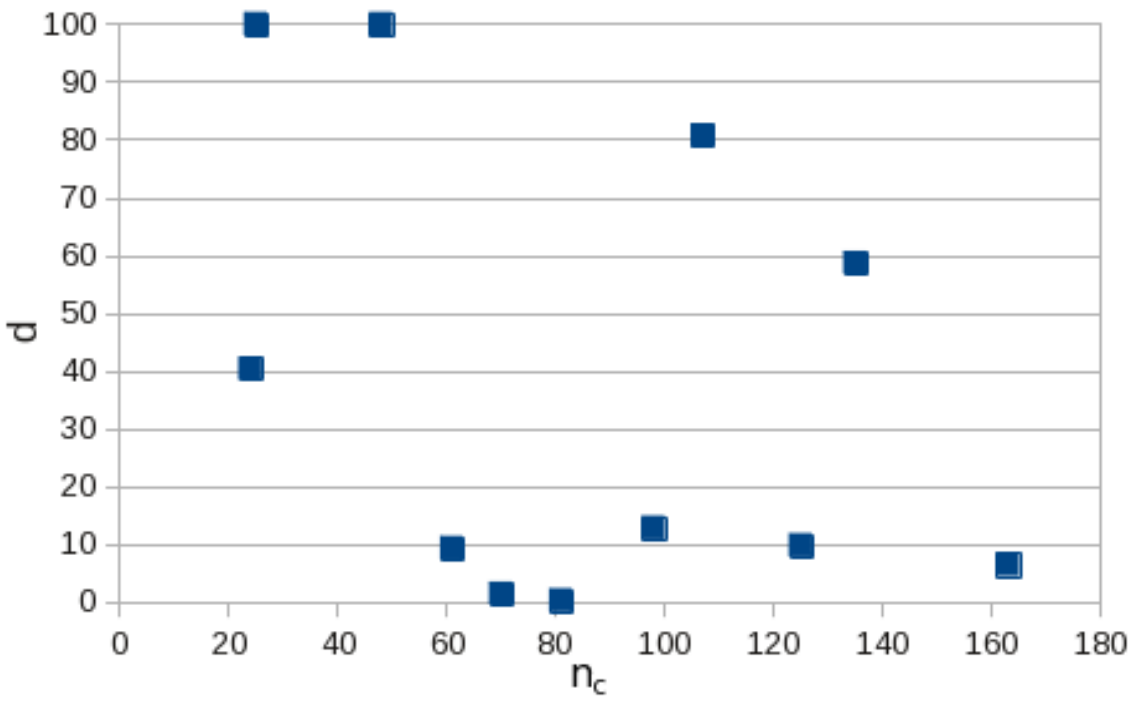


Figure I.6 – Difficulté du problème généralisé selon n_c. On voit ici que la difficulté d'un problème est davantage liée à sa propre structure qu'à sa taille (compressée) n_c. Ces données viennent de l'application en bio-informatique.

Annexe II

Code, pseudo-code et données complémentaires au mémoire

Exemple détaillé de la compression exacte pour le cas ADHD_avecBI.

```
 ADHD_avecBI.txt

ensemble A (4) :
[[7],[3,2],[31,41,4,5,1],[8],[27,43],[42],[40],[6],[17],[38],[20,10,11,9,30,39,16,
  12,14,13,23,19,22,24],[21],[15],[29],[28,36,32,45,44,35,33,37],[34,25,18,26]]
[[7],[31,41,4,5,1,3,2],[8],[6,17],[38,42,27,43,15,40,29],[20,34,28,10,25,11,9,18,
  30,36,39,26,32,16,12,14,13,45,44,35,33,23,37,21,19,22,24]]
[[7],[31,41,4,5,1,2],[3],[27],[8],[42,6,43],[10,11,9,38,36,39,16,12,14,13,40,44,23,
  19,22],[17],[15,45,33,24],[20,30,32,29],[35,37,21],[28],[34,25,18,26]]
[[7],[3,2],[31,41,4,5,1],[8],[6],[27,43,17,40],[15,29],[34,25,18,38,42,26,21],[20,
  28,10,11,9,30,36,39,32,16,12,14,13,45,44,35,33,23,37,19,22,24]]

Compression exactes:
1 <-- 1, 4, 5, 31, 41
2 <-- 2
3 <-- 3
4 <-- 6
5 <-- 7
6 <-- 8
7 <-- 9, 10, 11, 12, 13, 14, 16, 19, 22, 23, 39
8 <-- 15
9 <-- 17
10 <-- 18, 25, 26, 34
11 <-- 20, 30
12 <-- 21
13 <-- 24
14 <-- 27
15 <-- 28
16 <-- 29
17 <-- 32
18 <-- 33, 45
19 <-- 35, 37
20 <-- 36, 44
21 <-- 38
22 <-- 40
23 <-- 42
24 <-- 43
Compression des egalites: 45 --> 24
```

```
ensemble A compresse exacte (4) :
[[5],[2,3],[1],[6],[24,14],[23],[22],[4],[9],[21],[7,11,13],[12],[8],[16],[17,19,
  18,20,15],[10]]
[[5],[1,2,3],[6],[4,9],[16,21,23,22,8,24,14],[17,19,18,20,7,10,11,12,13,15]]
[[5],[1,2],[3],[14],[6],[4,23,24],[21,20,22,7],[9],[18,8,13],[17,16,11],[19,12],
  [15],[10]]
[[5],[2,3],[1],[6],[4],[22,9,24,14],[16,8],[21,23,10,12],[17,19,18,20,7,11,13,15]]

SA:
distance mediane = 682
[[5],[2,3],[1],[6],[4],[14,24],[23],[22],[9],[21],[8,16],[7,11,13,15,17,18,19,20],
  [10,12]]

Decompression:
distance mediane = 682
[[7],[2,3],[1,4,5,31,41],[8],[6],[27,43],[42],[40],[17],[38],[15,29],[9,10,11,12,
  13,14,16,19,22,23,39,20,30,24,28,32,33,45,35,37,36,44],[18,25,26,34,21]]
```

La correspondance des entiers avec des gènes.

```
Maping des genes pour ADHD_avecBI.txt

1 ADRA2A
2 DRD4
3 DRD5
4 SLC6A3
5 SNAP25
6 TPH1
7 TPH2
8 ADHD
9 ADHD1
10 ADHD2
11 ADHD3
12 ADHD4
13 ADHD5
14 ADHD6
15 AMT
16 ASPG2
17 CES1
18 CES5A
19 DEL1Q21
20 DOCK3
21 DRD5P1
```

22 DUP1Q21
23 DUP22Q11.2
24 DYX2
25 ESA4
26 ESB3
27 FGD1
28 GCSH
29 GLDC
30 GTS
31 HTR1B
32 IMMP2L
33 KIAA0319
34 LIPA
35 LOC731788
36 MED12
37 NF1
38 RAI1
39 RLS
40 SCN8A
41 SLC6A2
42 SLC9A9
43 SLITRK1
44 TBX1
45 TMEM132E

Algorithm 3 Simulated Annealing($\mathscr{R}$, $nbPoints$, $nvMvts$, α)

Data: a multi-set $\mathscr{R}$ of rankings with ties, the number of starting points $nbPoints$, the number of movements allowed $nbMvts$, the percentage of fission movements α.

Result: The set $BestC$ containing the best consensus found i.e. the rankings r minimizing $K(r,\mathscr{R})$

```
begin
    n ⟵ number of elements in the rankings of ℛ;
    BestC ⟵ {} ;
    λ ⟵ 0.99 ;
    EMin ⟵ ∞ ;
    for i ← 1 to nbPoints do
        T ⟵ 100000 ;
        r ⟵ RandomRankings(n) ;
        E ⟵ K(r,ℛ);
        EMin ⟵ E;
        for j ← 1 to nbMvts do
            rand ⟵ random[0,1];
            if rand < α then
                neighbour ⟵ mvtFission(r)
            else
                neighbour ⟵ mvtFusion(r)
            endif
            ΔE ⟵ E − K(neighbour,ℛ);
            if ΔE ≤ 0 then
                r ⟵ neighbour ;
                E ⟵ E + ΔE
            else
                rand ⟵ random[0,1];
                if rand < e^(−ΔE/T) then
                    r ⟵ neighbour;
                    E ⟵ E + ΔE
                endif
            endif
        endfor
        if E < EMin then
            BestC ⟵ {r}
        endif
        if E = EMin then
            BestC ⟵ BestC ∪ {r}
        endif
        T ⟵ T * λ
    endfor
    return BestC;
end
```