Diviser n'est pas régner ?

Laurent Lyaudet*

29 mai 2022

Résumé

En première année d'algorithmie, on découvre des exemples de schémas « Diviser pour régner » où une bonne division de l'instance du problème algorithmique étudié fournit aussi un algorithme de résolution efficace. Dans cet article, je présente un moyen de diviser efficacement tous les graphes de degré borné. Ce qui amène au fait intéressant que si $\mathsf{P} \neq \mathsf{NP}$, alors « Diviser n'est pas régner. ».

Version initiale : 2022/05/08 Version courante : 2022/05/29

1 Introduction

Depuis le début de l'algorithmique, une des méthodes les plus efficaces de résolution d'un problème est de diviser ou de décomposer une instance, puis de recombiner les solutions partielles des parties issues de la division/décomposition pour obtenir une solution globale (optimale). Trouver de bonnes manières de décomposer un problème est toujours un sujet de recherche très actif, notamment en théorie algorithmique et structurelle des graphes. On peut citer les décompositions modulaires, arborescentes, de branche, de rang, booléennes, etc. Il y en a pléthore et certaines sont à peu près équivalentes entre elles dans les instances qu'elles arrivent à décomposer.

Dans une première partie, je vous propose d'en découvrir une basée sur le principe de première différence que j'ai introduite en 2019 (Lyaudet (2019)) qui permet de décomposer tous les graphes de degré borné. Dans un second temps, nous verrons que cette décomposition amène naturellement à un algorithme de complexité « modérément exponentielle » pour le problème du stable maximum, même si cet algorithme n'est pas aussi performant que ceux de l'état de l'art pour le même problème. Le terme consacré pour des algorithmes de complexité inférieure à $O(2^n)$ dans la littérature est « subexponential », ce qui est assez fâcheux puisque la complexité de ces algorithmes dans le pire cas reste exponentielle, on a juste une base plus petite. Je préfère employer les termes de « modérément exponentielle » pour qualifier la complexité de ces algorithmes. Rappelons que le problème du stable maximum consiste à trouver un ensemble de sommets du graphe de cardinalité maximum, tel qu'aucune paire de sommets de cet ensemble n'est reliée par une arête. Le terme anglophone est « Maximum Independent Set ».

*https://lyaudet.eu/laurent/, laurent.lyaudet@gmail.com

2 Le principe de première différence et la largeur arborescente questionnable bijective équilibrée

Le principe de première différence est un principe très simple que tout le monde connaît sans le savoir, il a au moins 3000 ans comme la numération de position. L'exemple le plus direct est l'ordre du dictionnaire ; pour comparer et ranger deux mots on regarde la première lettre qui diffère de celle de l'autre mot à la position équivalente. Si cette lettre est plus petite que l'autre dans l'ordre des lettres, on range le mot avant, si elle est plus grande, on range le mot après. Par exemple, « première » vient avant « principe » car l'on a comparé la première différence entre le « e » et le « i ». En adhérant au second principe qu'une « lettre inexistante » vient toujours avant, on obtient l'ordre lexicographique utilisé dans le dictionnaire. On fait exactement la même chose pour comparer deux nombres dont les écritures dans une même base ont la même longueur. Par exemple, 1118 vient avant 1122 car le 1 est avant le 2. En mettant là aussi avant les écritures les plus courtes, on obtient cette fois pour les nombres l'ordre hiérarchique, puisque la longueur totale des écritures est considérée avant le principe de première différence.

J'ai proposées deux idées en conclusion de Lyaudet (2019) :

— on peut utiliser le principe de première différence non seulement pour définir des relations d'ordre mais aussi d'autres types de relations binaires, comme par exemple l'adjacence entre sommets dans les graphes ; la première différence entre les caractères associés à deux sommets déterminera si ceux-ci sont adjacents ou non ;

— on peut étendre le principe de première différence entre deux suites de caractères à un principe de première différence entre des caractères disposés sur un arbre et non un chemin.

Voyons ce que cela donne dans le cas des graphes.

Soit un ensemble de sommets V, une (V, k)-*suite-d'applications* est une suite d'applications (fonctions totales au sens mathématique) de V vers les sommets de graphes de cardinalité au plus k (un même graphe par application).

Definition 2.1. *Soit G un graphe. Une* (k, α, β)-décomposition arborescente questionnable bijective *de G est un triplet (A, ef, en) (A comme arbre, ef comme étiquetage des feuilles et en comme étiquetage des nœuds) :*

— *A est un arbre binaire enraciné ;*

— *les feuilles de A sont en bijection, par l'intermédiaire de la fonction ef, avec les sommets de G ;*

— *ainsi à chaque nœud interne $node$ est associé l'ensemble de sommets de G union des valeurs $ef(f)$ pour toutes les feuilles f sous le nœud $node$, ce qui définit $ef(node)$;*

— *en est une application ayant pour domaine les nœuds internes de A, telle que $en(node)$ est une $(ef(node), k)$-suite-d'applications,*

— *en conséquence, à chaque sommet de G correspond un sous-arbre (qui est un chemin dans cette définition simplifiée) de A, et puisque l'intersection de deux arbres, resp. chemins, est un arbre, resp. chemin, on a aussi un chemin correspondant à tout couple de sommets (x, y). On peut ainsi définir la $(\{x, y\}, k)$-*

suite-d'applications obtenue en concaténant les $(ef(node), k)$-suites-d'applications restreintes à $\{x, y\}$, et l'on impose que la première différence entre l'image de x et de y dans cette suite-d'applications existe et qu'elle corresponde à deux sommets adjacents, resp. non-adjacents, si x et y sont adjacents, resp. non-adjacents ;

— *α est la profondeur de l'arbre A ;*

— *β est la profondeur de l'arbre étendu A' obtenu en remplaçant chaque nœud interne par un chemin de nœuds (un pour chaque application de la suite associée au nœud original).*

k est appelé la largeur de la décomposition ; α est appelée la profondeur structurelle de la décomposition ; β est appelée la profondeur logique de la décomposition.

En fait cette définition est très puissante puisque tout est décomposable avec une largeur de 2 et une profondeur linéaire. Il suffit d'adjoindre les sommets à un arbre ressemblant à un peigne un par un.

Pour limiter cette puissance, on se restreint aux décompositions équilibrées, pour lesquelles la profondeur structurelle est logarithmique en la taille du graphe décomposé.

3 La décomposition des graphes de degré borné

Quand on cherche à décomposer un graphe, intuitivement, on essaye de mettre ensemble les parties les plus connectées entre elles. Sauf que là ça ne marche pas du tout ; il faut faire l'inverse. Si on a un graphe où le degré maximal d'un sommet est au plus d, disons 3 par exemple, il y a au plus $1 + d + d \times (d - 1) = 1 + d^2$, par exemple 10, sommets dans une boule de rayon 2 centrée sur un sommet. Donc on peut découper notre graphe en au plus $1 + d^2$, par exemple 10, parties pour lesquelles chaque sommet est au moins à distance 3 de tous les autres sommets de la même part. Il suffit d'un algorithme glouton : tant qu'un sommet n'est pas affecté à une des parts, je l'affecte à la première part qui ne contient pas d'autre sommet de la boule de rayon 2 centrée sur ce sommet.

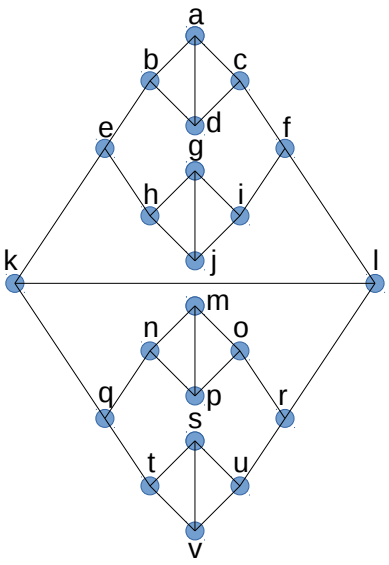
Du coup, dans la décomposition, on a un stable pour chaque partie, c'est-à-dire un sous-graphe induit sans arête, avec une décomposition modulaire ou questionnable triviale : on prend un arbre binaire équilibré où les feuilles sont en bijection avec les sommets de la partie et où chaque noeud interne est associé à une seule application qui envoie les sommets de gauche dans le sommet gauche et les sommets de droite dans le sommet droit d'un graphe à deux sommets non-adjacents, un à gauche, un à droite.

Comment est-ce que l'on relie les parties entre elles ? Avec la structure de peigne qui décompose tout. Du coup, on a un sous-graphe qui grossit de plus en plus car on lui ajoute les parties une à une, en ayant initialisé le sous-graphe par une partie quelconque. Supposons que ce sous-graphe est toujours à gauche et que la partie ajoutée est à droite. Comme les sommets de la partie ajoutée sont à distance au moins 3, aucun sommet du sous-graphe à gauche n'est adjacent à plus d'un sommet de la partie droite. On a donc un graphe biparti relativement trivial, tout comme sa décomposition bimodulaire, puisque c'est une union de sommets isolés et d'étoiles, une étoile en théorie des graphes est un graphe avec un sommet central qui est adjacent à des sommets

extérieurs et aucune des paires de sommets extérieurs n'est adjacente. En termes de suite d'applications, c'est quasiment aussi trivial qu'à l'intérieur des parties. On commence par faire l'astuce du graphe à deux sommets non-adjacents, un à gauche, un à droite, un nombre logarithmique de fois, pour décomposer le graphe biparti selon ses composantes connexes (les sommets isolés et les étoiles). Et pour couronner le tout, on fait une application vers un graphe à deux sommets adjacents, un à gauche, un à droite, un pour le sous-graphe de gauche et un pour la partie de droite. Par application du principe de première différence, les adjacences définies correspondent aux étoiles et aux sommets isolés, les adjacences intra-parties ayant déjà été fixées auparavant. On obtient une décomposition de largeur 2 dont les profondeurs structurelles et logiques sont logarithmiques.

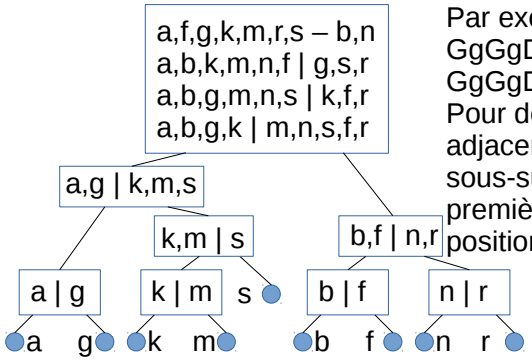
Il est aisé d'encoder les informations liées à un sommet avec 4 caractères : g ou d sommet gauche ou droit d'une application, G ou D branche de gauche ou de droite dans la décomposition, voire 6 avec g' et d' si on veut distinguer les applications qui envoient vers deux sommets adjacents de celles qui envoient vers deux sommets non adjacents. (L'option avec 6 caractères donne une information redondante puisque l'image d'une application ne dépend pas d'un sommet mais de la position dans l'arbre de décomposition. De plus, seule la dernière application de la réunion de deux parties a pour image deux sommets adjacents.) Une fois que l'on a les 2 suites de caractères associées à deux sommets, on part de la fin des deux suites (la racine de l'arbre de décomposition), et on garde tous les caractères avant tant qu'il n'y a pas de différence entre deux majuscules (G ou D). Ainsi, on sélectionne le sous-chemin commun aux deux sommets dans l'arbre de décomposition. Ensuite, il n'y a plus qu'à regarder la première différence entre ces deux sous-suites qui sont nécessairement de même longueur.

Exemple de graphe 3-régulier :



L'algorithme glouton sur les sommets dans l'ordre des lettres nous donne les 6 parties suivantes : (a,g,k,m,s), (b,f,n,r), (c,e,o,q), (d,h,l,p,t), (i,u), (j,v).

Exemple de début de décomposition correspondant aux 2 premières parties où la première différence s'évalue en partant du bas de la décomposition et en remontant vers la racine : on représente avec la barre verticale '|', respectivement le tiret, quand on associe les sommets à deux sommets non adjacents, resp. adjacents.
Il est aisé d'encoder les informations liées à un sommet avec 4 caractères : g ou d sommet gauche ou droit d'une application, G ou D branche de gauche ou de droite.



Par exemple, k donne GgGgDdGgdgg, b donne GgGgDgggd.
Pour déterminer, leur adjacence, on garde les sous-suites gdgg et gggd, la première différence est en position 2 sur deux sommets non adjacents donc k et b sont non adjacents.

FIGURE 1 – Exemple de décomposition

4 Une application simple au problème de stable maximum

Le problème du stable maximum reste NP-complet sur les graphes de degré 3. La réduction depuis 3-SAT est assez simple. Déjà sans restreindre le degré, on peut associer chaque clause à un triangle (un graphe à 3 sommets tous reliés), où chaque sommet du triangle correspond à un des littéraux de la clause. De plus chaque littéral est adjacent aux littéraux correspondant à sa négation présents dans les autres clauses. De cette manière, il est impossible d'avoir un littéral et une de ses négations dans un stable maximum. Comme on ne peut pas avoir plus d'un sommet par triangle, c'est-à-dire un littéral par clause, un stable maximum ne peut avoir plus de sommets que le nombre de clauses, et il n'atteint ce nombre que s'il correspond à une affectation de valeurs de vérité rendant vraie au moins un littéral par clause. Pour avoir un degré au plus 3, puisque le degré dans chaque triangle est de 2, au lieu de relier directement chaque littéral à ses négations, on relie les littéraux portant sur la même variable booléenne par un cycle de longueur paire. Pour chaque cycle, on fixe une origine et on associe les littéraux positifs à l'un des sommets de rang pair depuis l'origine, et les littéraux négatifs à l'un des sommets de rang impair depuis l'origine. Comme un stable dans un cycle pair ne peut couvrir qu'un sommet sur deux et que le stable maximum est atteint si l'on prend tous les sommets pairs ou bien tous les impairs, la synchronisation est assurée. Et là encore le maximum n'est atteint que si l'on s'est abstenu de prendre deux littéraux contraires.

On a donc une décomposition en 10 parties pour le problème du stable maximum dans les graphes de degré 3. Trouver un stable maximum dans une partie, c'est trivial puisque chaque partie est un stable. Dans l'union de deux parties, ce n'est pas beaucoup plus compliqué puisque l'on a des sommets isolés et des arêtes isolées, donc il suffit de prendre les sommets isolés et un sommet par arête isolée. Dans l'union de trois parties, chaque sommet est de degré au plus deux donc on a une union de cycles et de chemins, possiblement réduits à un sommet isolé, donc là aussi c'est trivial. Et alors pour 4 parties, c'est-à-dire la troisième fois que l'on réunit des parties ? Là ça devient NP-complet. Ah bon déjà, on n'attend même pas la fin des réunions de parties ? ;)

Pour montrer cela, on va d'abord voir que le problème reste NP-complet sur un problème encore moins dense que le précédent. Comment est-ce que l'on peut faire moins dense ? Avant 3, c'est 2, on ne va tout de même pas se ramener à des chemins ou des cycles ? Presque, on va remplacer chaque arête par un chemin de longueur 3, on aura donc créé deux sommets intermédiaires entre deux sommets du graphe d'origine. Tout d'abord on peut remarquer qu'en faisant cela on augmente la taille d'un stable maximum d'au plus le nombre d'arêtes. En effet, le nombre maximum de sommets choisis dans le stable maximum sur un chemin de longueur 3, c'est toujours un plus le nombre de sommets choisis aux extrémités quitte à déplacer un choix. Pour être convaincu, c'est plus simple de s'imaginer remplacer les arêtes par un chemin une à une, en obtenant autant de graphes intermédiaires que d'arêtes et de voir que la cardinalité du stable maximum n'augmente que d'un à chaque fois. Nommons $x - r - s - y$ les sommets du chemin, dans un sens :

— sachant que j'ai choisi x et pas y dans un stable maximum du graphe de départ,

je peux ajouter s et j'obtiens bien plus 1, je ne peux pas faire plus localement,
- sachant que j'ai choisi y et pas x dans un stable maximum du graphe de départ, je peux ajouter r et j'obtiens bien plus 1, je ne peux pas faire plus localement,
- sachant que je n'ai choisi ni x ni y dans un stable maximum du graphe de départ, je peux ajouter r ou s et j'obtiens bien plus 1, x et y étaient chacun bloqués par un autre sommet du graphe, je ne peux pas faire plus localement ;

dans l'autre sens :

- sachant que j'ai choisi x et y dans un stable maximum du graphe d'arrivée, je remplace arbitrairement y par s et j'obtiens bien moins 1 en revenant au graphe d'arrivée, je ne peux pas faire plus localement,
- sachant que j'ai choisi x et s, resp. y et r dans un stable maximum du graphe d'arrivée, j'obtiens bien moins 1 en revenant au graphe de départ, je ne peux pas faire plus localement,
- sachant que j'ai choisi r, resp. s, dans un stable maximum du graphe d'arrivée, j'obtiens bien moins 1 en revenant au graphe de départ, je ne peux pas faire plus localement.

En notation synthétique, à la probabilité conditionnelle,

$$\begin{aligned}
StableMax(G) = max(&\\
&StableMax(G|x, \neg y),\\
&StableMax(G|\neg x, y),\\
&StableMax(G|\neg x, \neg y)\\
)&\\
= max(&\\
&StableMax(G'|x, \neg y) - 1,\\
&StableMax(G'|\neg x, y) - 1,\\
&StableMax(G'|\neg x, \neg y) - 1,\\
&StableMax(G'|x, y) - 1\\
)&\\
= StableMax(G') - 1&
\end{aligned}$$

car $StableMax(G'|x, \neg y) \geq StableMax(G'|x, y)$ par exemple.

Bon c'est sympa cette réduction à un graphe encore moins dense, mais il reste 3 voisins directs puis 3 voisins de plus à chaque sommet du graphe de départ. Donc les sommets du graphe de départ sont dans une boule de taille au plus 7, et les nouveaux sommets dans une boule de taille au plus 6. On est encore loin de la décomposition en 4 parties. On va voir tout de suite qu'en temps polynomial, on peut agencer tout ça pour faire mieux. On va commencer par mettre tous les sommets du graphe original dans une partie, puisque maintenant ils sont à distance 3. Il nous reste les sommets de degré deux adjacents aux sommets originaux. Il nous reste 3 étapes, 3 parties et on se retrouve dans un problème d'orientation d'arêtes. Choisir le sommet r, c'est un peu comme choisir l'arête $x - y$ dans ce sens là, alors que choisir s c'est choisir l'arête $y - x$ dans ce sens ci. Du coup, on ne peut pas choisir à la fois r et s dans la même étape/partie. De plus, on ne peut pas non plus choisir deux arcs partant du même sommet, car cela revient à

choisir deux sommets à distance 2 autour du sommet. En dehors de ces contraintes, on est libre de notre choix puisque les autres sommets restants sont à distance au moins 3.

Procédure de choix pour la deuxième partie : Pour chaque sommet du graphe d'origine, on prend un arc sortant de ce sommet (matérialisé par un des nouveaux sommets). Si cet arc est en opposition avec un autre arc, on en prend un autre sauf si les 3 arcs rentrent en opposition, auquel cas, on renverse l'arc en opposition en le basculant sur un des 2 autres arcs partant de l'autre sommet ; et on itère le renversement sur l'un des deux autres arcs si les deux autres arcs sont en opposition ; tant qu'on est obligé de renverser, cela signifie que tous les sommets vus jusqu'alors étaient orientés vers le sommet qu'on cherchait à satisfaire via le chemin de renversement courant ; le nombre de sommets étant fini, on arrive forcément au bout du chemin à un sommet qui n'a pas d'opposition à un renversement éventuel. Cela prend un temps au plus quadratique en le nombre de sommets.

Procédure de choix pour la troisième partie : Elle est très similaire à la deuxième partie. Pour chaque sommet du graphe d'origine, on prend un arc sortant de ce sommet (matérialisé par un des nouveaux sommets) non sélectionné à l'étape 2. Si cet arc est en opposition avec un autre arc, on en prend un autre sauf si les 2 arcs non sélectionnés rentrent en opposition, auquel cas, on renverse l'arc en opposition en le basculant sur l'autre arc non sélectionné partant de l'autre sommet ; et on itère le renversement sur l'autre arc non sélectionné si l'autre arc est en opposition ; tant qu'on est obligé de renverser, cela signifie que tous les sommets vus jusqu'alors étaient orientés vers le sommet qu'on cherchait à satisfaire via le chemin de renversement courant ; le nombre de sommets étant fini, on arrive forcément au bout du chemin à un sommet qui n'a pas d'opposition à un renversement éventuel. Cela prend un temps au plus quadratique en le nombre de sommets.

Procédure de choix pour la quatrième partie : Si toutes les arêtes ont eu au moins une orientation dans la deuxième ou troisième étape, il suffit de prendre le dernier arc sortant de chaque sommet et l'on n'aura pas de conflit. Sinon, on a une arête dont les deux sommets incidents souhaitent la sélectionner à la quatrième étape. On décide de privilégier le sommet x au sommet y à l'étape 4 pour l'arc $x \to y$. Le sommet y doit donc prendre l'arc $y \to x$ dans la troisième partie, on va donc récrire un choix d'une étape précédente. Ce qui correspond à une nouvelle suite de renversements dans l'étape 3. Allant vers y, il y a à l'étape 4 jusqu'à présent un arbre orienté vers y. Partant de y, il y a à l'étape 3 un chemin orienté correspondant aux choix de cette étape avant réécriture, de plus ce chemin se termine nécessairement par un sommet qui a choisi un arc allant vers un sommet précédent du chemin, là encore par finitude. Si l'on prend l'intersection de cet arbre et de ce chemin, cela nous donne un chemin. On peut renverser les choix de ce chemin dans la troisième et dans la quatrième partie. Les orientations ne posent aucun problème aux étapes 3 modifiée et étape 4 pour le sommet x, les sommets intérieurs au chemin (intersection) précisé ayant y pour extrémité sauf l'extrémité opposée à y. Si le sommet opposé à y sur le chemin, éventuellement y si le chemin est réduit à un sommet, n'est pas le sommet final du chemin partant de y à l'étape 3, il est évident que l'on peut lui donner à l'étape 4, l'arc qui allait vers son successeur à l'étape 3 puisque ce successeur n'est pas dans l'arbre allant vers y de l'étape 4. Sinon, cela signifie que l'arbre de l'étape 4 allant vers y contient exactement l'inverse du chemin orienté de l'étape 3. Sauf que cet arbre ne peut boucler puisque chaque sommet

n'a qu'un arc sortant. Donc le voisin sortant pour cet ultime sommet dans l'étape 3 est aussi un voisin sortant possible pour cet ultime sommet dans l'étape 4. Là encore, ce jeu de renversements prend un temps au plus quadratique en le nombre de sommets.

On a donc montré que trouver un stable maximum dans la réunion de 3 parties était trivial, alors que cela devient NP-complet avec 4 parties. Pour l'algorithme de complexité « modérément exponentielle » promis, il suffit de voir que le graphe moins dense a exactement $4n$ sommets, si on avait n sommets au départ. Chacune des parties a exactement n sommets. La réunion des 3 premières parties est une union de cycles et de chemins, car tout sommet y est de degré au plus deux. Mais le point important, c'est que l'on n'a pas d'orientation contradictoire à la quatrième étape. Donc pour tout sommet original, qui est nécessairement de degré 2 dans l'union des 3 premières parties, il existe un sommet original orienté vers lui dès la deuxième ou troisième étape, c'est à dire une extrémité de chemin. Donc il y a exactement $n/2$ chemins et donc au moins $n/2$ composantes connexes, chaque composante étant un cycle ou un chemin. Pour la troisième réunion entre les 3 premières parties et la quatrième, si l'on considère d'abord au moins $n/2 - 1$ sommets qui connectent tous les morceaux, on obtient un graphe n'ayant pas plus de deux chemins sommets disjoints entre deux sommets donnés. Un peu comme un arbre avec des cycles par endroits, par exemple une haltère (deux cycles reliés par un chemin). Trouver un stable maximum dans ces graphes là est facile, l'algorithme glouton qui consiste à voir un cycle qui n'est adjacent qu'à un seul chemin que comme un gros sommet, un cycle-feuille, nous garantit de toujours avoir une vraie feuille ou un cycle-feuille à traiter. Il est facile de voir qu'il existe toujours un stable maximum qui prend une feuille donnée, donc on la prend, on enlève son sommet voisin, et on itère. De même, si l'on a un cycle-feuille, il existe forcément un stable maximum qui est optimal localement sur ce cycle et qui ne prend pas le sommet adjacent au chemin adjacent au cycle-feuille. Donc on prend ce stable maximum local, on enlève le cycle-feuille et on itère. Au final, on fait une énumération exhaustive des stables maximum sur les $n/2 + 1$ sommets restants de la quatrième partie, et pour chaque choix de l'énumération, on calcule l'algo glouton sur ce qui reste en ayant enlevé les sommets adjacents aux sommets choisis dans le choix de l'énumération. On obtient un algorithme de complexité $O(2^{n/2+1})$ dont la complexité polynomiale masquée par la notation asymptotique est quadratique.

5 Conclusion

On voit que des principes très anciens des mathématiques, un peu remis au goût du jour, donnent des résultats sympathiques. Même si l'état de l'art sur les algorithmes modérément exponentiels pour le problème du stable maximum est plus performant, même dans le cas où le graphe n'est pas de degré borné. Pour d'autres résultats autour du principe de première différence, j'invite le lecteur intéressé à regarder les articles de Sierpiński et d'Hausdorff en théorie des ordres, la vaste littérature autour de la décomposition modulaire et plus modestement mes 3 articles sur le sujet (2018, 2019 et 2020), dont les deux premiers sont disponibles sur arXiv et le troisième uniquement sur ma page web.

Quant à mon interprétation théologique que sans doute Diviser n'est pas régner, je

dirai juste que quand bien même il y aurait un algorithme quantique polynomial pour ce problème, avec tous ces enchevêtrements, l'algorithme quantique *relierait* les parties du tout. Et j'y verrais encore l'occasion de dire :

Merci Dieu ! Merci Père ! Merci Seigneur ! Merci Saint Esprit !

Références

B. Bui-Xuan, M. Habib, V. Limouzy, and F. de Montgolfier. Homogeneity vs. adjacency : Generalizing some graph decomposition algorithms. In F. V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, volume 4271 of *Lecture Notes in Computer Science*, pages 278–288. Springer, 2006. doi : 10.1007/11917496_25. URL https://doi.org/10.1007/11917496_25.

B.-M. Bui-Xuan. *Tree-Representation of Set Families in Graph Decompositions and Efficient Algorithms*. PhD thesis, Université Montpellier II, september 2008.

G. Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Math. Ann.*, 46 : 481–512, 1895.

A. Ehrenfeucht, T. Harju, and G. Rozenberg. *The Theory of 2-Structures - A Framework for Decomposition and Transformation of Graphs*. World Scientific Publishing Co. Pte. Ltd., 1999. ISBN 981-02-4042-2.

J. Fouquet, M. Habib, F. de Montgolfier, and J. Vanherpe. Bimodular decomposition of bipartite graphs. In J. Hromkovic, M. Nagl, and B. Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, volume 3353 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2004. doi : 10.1007/978-3-540-30559-0_10. URL https://doi.org/10.1007/978-3-540-30559-0_10.

T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18 :25–66, 1967.

M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.*, 4(1) :41–59, 2010. doi : 10.1016/j.cosrev.2010.01.001. URL https://doi.org/10.1016/j.cosrev.2010.01.001.

F. Hausdorff. Untersuchungen über Ordnungstypen V. *Ber. über die Verhandlungen der Königl. Sächs. Ges. der Wiss. zu Leipzig. Math.-phys. Klasse*, 59 :105–159, 1907.

L. Lyaudet. A class of orders with linear ? time sorting algorithm. *CoRR*, abs/1809.00954, 2018. URL http://arxiv.org/abs/1809.00954.

L. Lyaudet. On finite width questionable representations of orders. *CoRR*, abs/1903.02028, 2019. URL http://arxiv.org/abs/1903.02028.

L. Lyaudet. First difference principle applied to modular/questionable-width, clique-width, and rank-width of binary structures. *preprint*, 2020. URL `https://lyaudet.eu/laurent/Publi/Journaux/LL2020LargeursStructuresBinaires/LargeursStructuresBinaires_v5.pdf`.

W. Sierpiński. Généralisation d'un théorème de Cantor concernant les ensembles ordonnés dénombrables. *Fundamenta Mathematicae*, 18 :280–284, 1932.