Diviser n'est pas régner ?

Laurent Lyaudet

November 19, 2023

Objectif de cette présentation : montrer comment diviser/décomposer efficacement (en temps quasi-linéaire) tous les graphes de degré borné.
Conséquence intéressante : si P $\neq$ NP, alors “Diviser n’est pas régner.”.

Tout le monde connaît le principe de première différence sans le savoir.
Comparer avec ce principe :

- deux mots : “première” $<$ “principe” car $p = p$, $r = r$, $e < i$;
- deux nombres : $1118 < 1122$ car $1 = 1$, $1 = 1$, $1 < 2$.

De plus :

- deux mots : “pre” vient avant “première” car $p = p$, $r = r$, $e = e$ et “pre” est plus court : ordre lexicographique ;
- deux nombres : 321 vient avant 1122 car 321 est plus court : ordre hiérarchique.

Ce principe a au moins 4000 ans comme la numération de position. Il devrait faire partie de la terminologie de tout(e) honnête scientifique.

Hausdorff avec -1, 0, 1 et Sierpiński avec 0, 1:

Théorème (Hausdorff 1907, Sierpiński 1932)

À chaque élement de tout ordre total de cardinal $\aleph$, on peut associer une suite ternaire/binaire de longueur au plus $\alpha(\aleph)+1$, où $\alpha(\aleph)$ est le premier ordinal de cardinal $\aleph$, de telle sorte que la comparaison des suites par le principe de première différence donne l'ordre total considéré.

Hausdorff 1907 : Untersuchungen über Ordnungstypen V
Sierpiński 1932 : Généralisation d'un théorème de Cantor concernant les ensembles ordonnés dénombrables

J'appelle la première différence entre deux suites "la question".
Comparer avec ce principe :

- (être, être, être, être) et
- (être, être, être, ne pas être).

On peut considérer des structures avec des fonctions ou relations binaires arbitraires.
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).

Définition (Lyaudet 2019)

Soit G un graphe. Une (k, β)-décomposition questionnable de G est une $(V(G), k)$-suite-d'applications de longueur β, telle que, pour toute paire de sommets $x, y \in V(G)$, 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. k est appelée la largeur de la décomposition ; β est appelée la profondeur de la décomposition.

La largeur questionnable est équivalente à la largeur modulaire (Lyaudet 2020 : First difference principle applied to modular/questionable-width, clique-width, and rank-width of binary structures).

Lyaudet 2019 : On finite width questionable representations of orders.
Deux idées dans la conclusion de cet article :

- le principe de première différence convient aux ordres, aux graphes et aux structures binaires en général ;
- 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.

Définition (Lyaudet 2019)

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 $ef(node)$;*

Définition (Suite...)

- *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/chemin de A, et puisque l'intersection de deux chemins est un 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 ;*

Définition (Suite 2...)

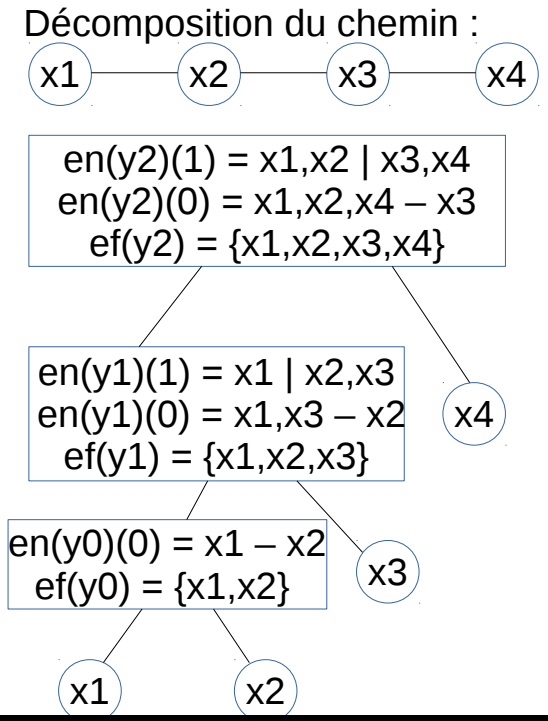
- *α est la profondeur de l'arbre A ;*
- *β est la profondeur de l'arbre étendu A' obtenu en remplaçant chaque nœud interne par un chemin de nœuds (un pour chaque application de la suite associée au nœud original).*

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

Trop puissant:

Lemme (Lyaudet 2019)

Tout graphe (structure binaire) a une $(2, \alpha, \leq \beta)$-décomposition arborescente questionnable bijective, où α est le premier ordinal de même cardinal que V, et β est un ordinal de même cardinal que V^2.



"Solution" : dans le cas fini, on peut se restreindre aux décompositions équilibrées : profondeur structurelle logarithmique.

Largeur/esse à tout faire ;)

Lemme (Lyaudet 2019)

Si un graphe a une décomposition arborescente binaire de largeur k et profondeur d, il a une $(k+2, d+1, d)$-décomposition arborescente questionnable.

Lemme (Lyaudet 2019)

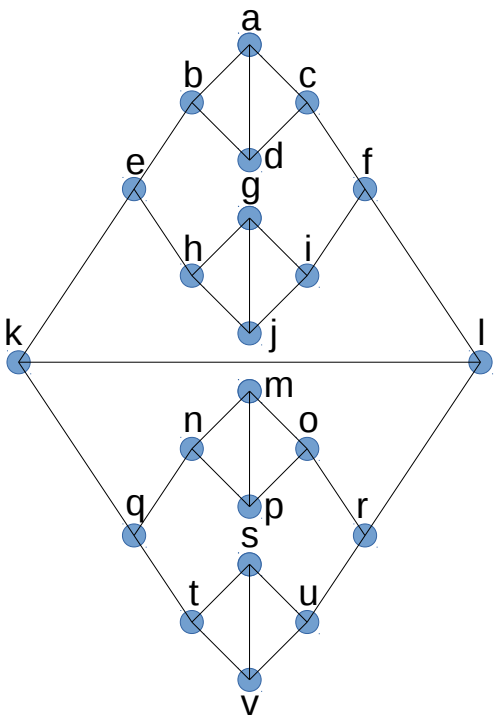
Si un graphe a une décomposition de clique compacte de largeur k et profondeur d, il a une $(2k, d, d-1)$-décomposition arborescente questionnable bijective.

Lemme (Lyaudet confinement 2020 non publié)

Si un graphe a une décomposition arborescente binaire de largeur k et profondeur d, il a une $(2, \leq (k \times d) + 1, \leq 2(k \times d))$-décomposition arborescente questionnable bijective.

En fait ce qu'il se passe sur chaque nœud interne correspond en gros à une décomposition bimodulaire : J. Fouquet, M. Habib, F. de Montgolfier, and J. Vanherpe 2004 : Bimodular decomposition of bipartite graphs.
Sauf qu'au lieu d'avoir une non-adjacence entre deux sommets d'une même partie, on a une "adjacence" **alf** (already fixed).

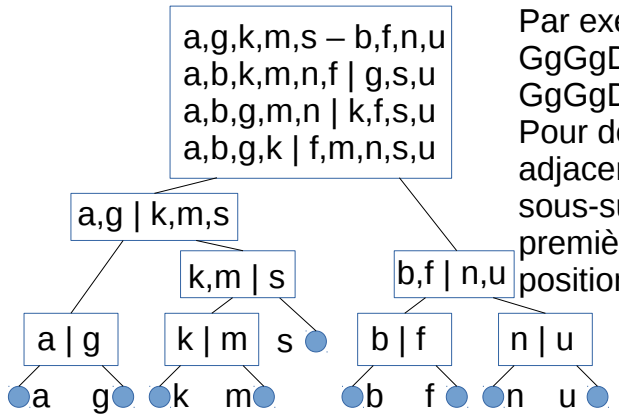
Graphes de degré borné : on veut découper en parties de telle sorte que les graphes "bipartis" de recollement soient le plus simple possible.
On prend des parties où tous les sommets sont à distance au moins 3. Comme ça, le recollement d'une partie correspond à une union d'étoiles (éventuellement dégénérées : sommet ou arête isolés).
Chaque partie est a fortiori un stable donc un arbre binaire équilibré avec des applications/compositions parallèles suffit.
Si on a un graphe de degré maximal d, disons 3 par exemple, il y a au plus $1 + d^2$, par exemple 10, sommets dans une boule de rayon 2 centrée sur un sommet. Donc on peut découper en au plus $1 + d^2$, par exemple 10, parties "3-espacées". Il suffit d'un algorithme glouton : tant qu'un sommet n'est pas affecté à une des parts, on l'affecte à la première part qui ne contient pas d'autre sommet de la boule de rayon 2 centrée sur ce sommet.



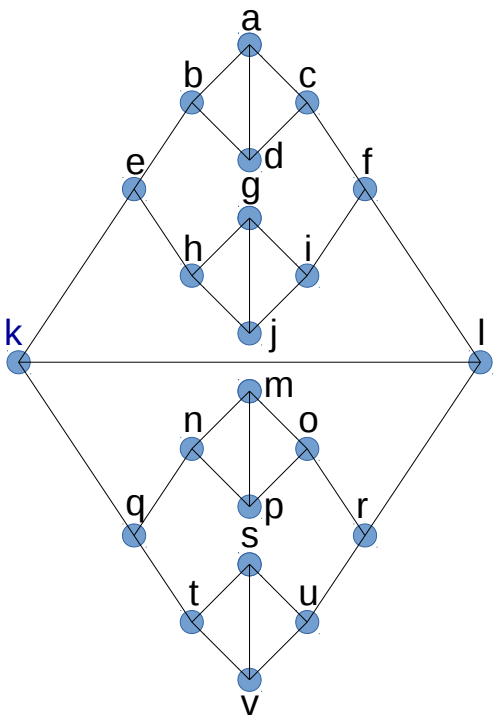
Exemple de graphe 3-régulier :

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,u), (c,e,o,t), (d,h,l,p,v), (i,q,r), (j).

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.



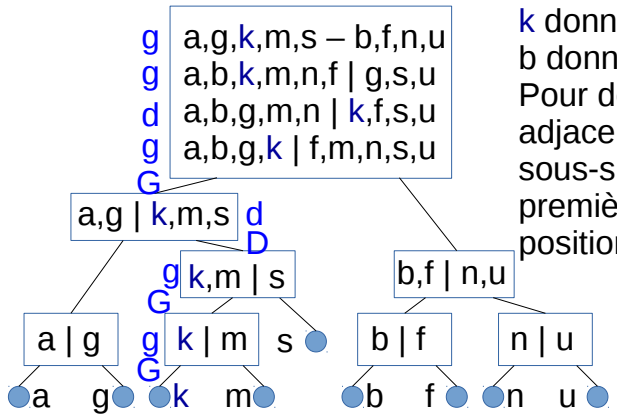
Par exemple, k donne GgGgDdGgdgg, b donne GgGgDgggd.
Pour déterminer, leur adjacence, on garde les sous-suites gdgg et gggd, la première différence est en position 2 sur deux sommets non adjacents donc k et b sont non adjacents.



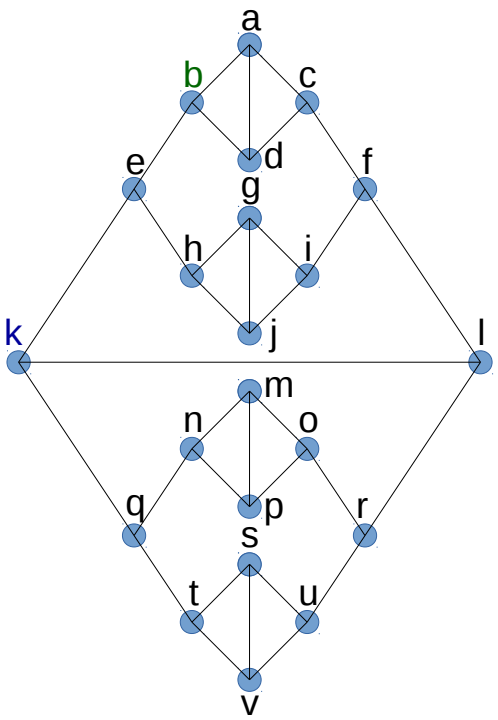
Exemple de graphe 3-régulier :

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,u), (c,e,o,t), (d,h,l,p,v), (i,q,r), (j).

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 :



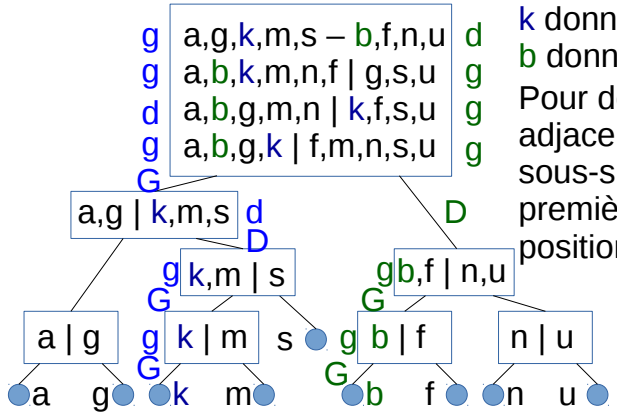
k donne GgGgDdGgdgg, b donne GgGgDgggd. Pour déterminer, leur adjacence, on garde les sous-suites gdgg et gggd, la première différence est en position 2 sur deux sommets non adjacents donc k et b sont non adjacents.



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,u), (c,e,o,t), (d,h,l,p,v), (i,q,r), (j).

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.

Théorème (Décomposition des graphes de degré borné par le principe de première différence, Lyaudet confinement 2020)

Tout graphe de degré au plus d à n sommets admet une décomposition arborescente questionnable bijective équilibrée de largeur 2, de profondeur structurelle au plus $\lceil \lg(n) \rceil + d^2$ et de profondeur logique au plus $\lceil \lg(n) \rceil + d^2 \times (\lceil \lg(n) \rceil + 1) = \lceil \lg(n) \rceil \times (1 + d^2) + d^2$. De plus, cette décomposition est calculable en temps quasi-linéaire.

Comme le problème du stable maximum est NP-complet sur les graphes de degré maximum 3, résoudre ce problème sur une union de 10 parties "3-espacées" est NP-complet. Mais on peut faire mieux : résoudre ce problème sur une union de 4 parties "3-espacées" est NP-complet. Alors que pour au plus 3 parties "3-espacées" c'est quadratique en temps. Voir les détails dans mon article Diviser n'est pas régner ? (2022) sur mon site, menu :

- une réduction vers des graphes encore moins denses,
- coloration propre du "carré" (2-distance coloring optimal non connu),
- une classe de graphe dont le "carré" est non parfait,
- mais, en enlevant une partie simple, on obtient une sous-classe des graphes parfaits (classification exacte en cours),
- algo linéaire de découpage en 4 parties,
- algo modérément exponentiel pour le problème du stable maximum.

Théorème (Décomposition des graphes de degré borné par la tree-tree-width, Lyaudet confinement 2020)

Tout graphe de degré au plus d à n sommets admet une décomposition arborescente arborescente (pas une typo) de largeur 1, de profondeur structurelle au plus $\lceil \lg(n) \rceil + d^2$. De plus, cette décomposition est calculable en temps quasi-linéaire.

Conclusion :

- Diviser les graphes de degré borné en temps quasi-linéaire avec une décomposition du second ordre, c'est facile.
- Que faire avec ?
- Démontrer que "S.E.T.H. is false"? Je n'ai pas (encore ;)) réussi. Mes résultats sont bien plus modestes :).
- Démontrer que plein de problèmes simples peuvent être résolus en temps quasi-linéaire ?

Conclusion 2 :

- Démontrer que plein de problèmes simples peuvent être résolus en temps quasi-linéaire ?
- Justement, en 2022, Maximum Flow and Minimum-Cost Flow in Almost-Linear Time (!!!), Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva.
- 109 pages, une petite thèse en guise de preprint arXiv. Qui l'a lu en entier et peut l'expliquer à ses étudiants ?
- Max-flow, c'est trivial à réduire en temps linéaire vers les graphes de degré borné (séparer les sommets de degré > 3 en sous-arbres avec des arcs dans les deux sens de capacité égale à la somme des capacités des arcs incidents).
- Pas sûr que la décomposition arborescente questionnable bijective convienne, mais peut-être qu'une décomposition bien choisie permettrait de simplifier et généraliser ce résultat.

Conclusion 3 :

- Justement, en 2022, Maximum Flow and Minimum-Cost Flow in Almost-Linear Time (!!!), Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva.
- 109 pages, une petite thèse en guise de preprint arXiv. Qui l'a lu en entier et peut l'expliquer à ses étudiants ?
- Justement, en 2020, Integer multiplication in time O(n log n) (!!!), David Harvey, Joris van der Hoeven.
- 45 pages. Qui l'a lu en entier et peut l'expliquer à ses étudiants ?

Conclusion 4 :

- Comment motiver nos étudiants si on dit à plus de 95 % d'entre eux "Dans 5 ans, tu auras progressé mais pas assez pour comprendre les algorithmes les plus évolués pour les problèmes de base comme la multiplication d'entier ou le flow-maximum." ?
- Moralité : pour le niveau qui baisse à l'école et les smartphones et autres écrans dans les mains des jeunes enfants, l'impact des scientifiques est limité. Mais faire en sorte qu'il n'y ait pas un précipice qui se creuse entre la recherche de haut niveau et même vos bons étudiants, ça veut dire mâchonner ces résultats jusqu'à les rendre digestes.
- Pas sûr que la course aux publications et au nom propre devant les complexités asymptotiques incite à aller dans cette voie.

Bonus : il existe une variante non arborescente du principe de première différence équivalente à la clique-width. Elle utilise une notion duale de l'"adjacence" **alf**, l'"adjacence" **nyf** (not yet fixed).

Théorème (Lyaudet 2020)

La largeur de clique questionnable d'un graphe (structure binaire) de cardinal dénombrable est entre sa largeur de clique et deux fois sa largeur de clique.

Mais contrairement à la largeur de clique, elle est définie pour des structures binaires de tout cardinal. Trouver un algorithme polynomial pour la calculer donnerait enfin une bonne approximation de la largeur de clique.