Factoriser des nombres entiers avec des graphes et autres idées liées

Laurent Lyaudet*

1er août 2022

Résumé

Dans cet article, je propose d'aborder le problème de la factorisation des nombres entiers comme un problème de comptage, puis de factoriser ou plus généralement d'avoir un modèle de calcul à base de graphes et d'opérations matricielles non triviales. Enfin, je présente une heuristique de factorisation amusante nommée Fibolous. Attention : expression libre et style non-académique sur la forme.

Version initiale : 2022/06/05 Version courante : 2022/08/01

1 Compter pour factoriser

Je commence par une idée que j'ai eu quelques jours avant Pâques cette année. Le problème de décision de la primalité d'un nombre entier peut être résolu en temps polynomial. On le sait depuis l'article de Agrawal et al. (2004). Malgré cela on ne sait rien de la complexité de factoriser un nombre entier. Peut-être que factoriser appartient à une autre classe de problèmes que les problèmes de décision, comme par exemple les problèmes de comptage des solutions. (En forçant, tout devient un problème de décision mais pas nécessairement en restant dans NP. Par exemple, compter les solutions d'un problème qui admet au plus un nombre exponentiel de solutions peut être fait par dichotomie en se posant un nombre linéaire de questions « Est-ce que mon instance a plus de k solutions ? » avec différentes valeurs de k. La démarche est similaire pour les problèmes d'optimisation.) Du coup, qu'est-ce que l'on pourrait compter comme solutions au problème de la factorisation d'un nombre n ? Comme le lecteur s'en apercevra en lisant les deux prochaines sections, la solution naturelle par rapport à ce que j'avais déjà essayé est de compter les nombres entiers inférieurs à n et qui ne sont pas relativement premiers avec n. Notons ce problème $\#\mathsf{NonRelativementPremiers}$. (Cette fonction de n a « presque » déjà un nom en mathématiques, mais faisons abstraction de ce point 5 minutes.)

Du point de vue de la stratégie de recherche, c'est un problème très intéressant, car il n'y a que 3 cas possibles :

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

— Si $\#\mathsf{NonRelativementPremiers}$ est $\#\mathsf{P}$-complet, alors comme on sait factoriser avec l'algorithme quantique de Shor (1997) et que compter ces nombres entiers inférieurs à n et qui ne sont pas relativement premiers avec n est trivial si je connais la factorisation, cela veut dire que les ordinateurs quantiques auraient encore plus une « puissance de malade » car ils résoudraient un problème $\#\mathsf{P}$-complet en temps cubique. Cette possibilité est tout à fait vraisemblable car il existe un exemple extrême où le problème de décision est trivial (toujours vrai) et le problème de comptage associé est $\#\mathsf{P}$-complet ; il s'agit de compter les extensions linéaires d'un ordre (Brightwell and Winkler (1991)).

— La possibilité opposée est que ce problème puisse être résolu en temps polynomial. Là encore ce serait un énorme résultat car on pourrait casser RSA en temps polynomial. En effet, si $n = pq$, il y a $p - 1 + q - 1$ nombres entiers inférieurs à n et qui ne sont pas relativement premiers avec n. Or si je connais $p + q - 2$, je connais $p + q$. Et si je connais pq et $p + q$, alors je connais p et q. (Programme de mathématiques du lycée, il y a 20–25 ans, on comprend mieux qu'ils allègent les programmes d'année en année, c'est plus efficace que de déclarer une guerre tout de suite, il suffit d'attendre 50 à 100 ans qu'il ne reste plus qu'une majorité de personnes ignares… Donc rappelons que si $n = pq$ et $p+q = k$, alors $p(p+q) = kp$, mais aussi $p(p+q) = p^2+pq = p^2+n$. Donc on a $kp = p^2 + n$, c'est à dire l'équation du second degré $p^2 - kp + n = 0$. Le discriminant c'est $k^2 - 4n$ et donc on a deux solutions : $\frac{k+\sqrt{k^2-4n}}{2}$ et $\frac{k-\sqrt{k^2-4n}}{2}$. Exemple avec 15 : $k = 8$, le discriminant c'est $64 - 60 = 4$, et on a bien les solutions $\frac{8+2}{2} = 5$ et $\frac{8-2}{2} = 3$… Bizarrement, mes cours de mathématiques de terminal ont été volés chez moi fin 2020 (pas de trace d'effraction, merveilleux, hein ? Si vous voulez vous rendre utile, commercialisez des vraies serrures au grand public. Vous pouvez regarder les vidéos youtube du LockPickingLawyer si ça vous intéresse.). Sans doute le privilége d'avoir dénoncé des espions quand j'étais en thèse. Mais bon, même si j'avais attendu d'être reconnu par mes pairs pour ouvrir ma gueule, je pense que cela ne m'aurait pas totalement protégé pour autant.)

— Soit enfin, ce problème définit sa propre classe de complexité intermédiaire, et on peut chercher des réductions depuis ou vers d'autres problèmes de comptage et définir une nouvelle classe de complexité intéressante.

On peut s'estimer gagnant dans au moins deux des trois cas. Personnellement, je trouve même le troisième cas intéressant. Comme quoi la première semaine de cours de logique et la disjonction de cas suffisent à stratégiser sa recherche…

Cette idée du problème de comptage, et notamment de l'implication « Savoir compter, c'est savoir casser RSA », m'est venue quelques jours avant Pâques 2022, alors que je ne me souciais plus trop de factorisation d'entiers. Je considère qu'elle vient de Dieu à cause du caractère soudain et non corrélé avec mon quotidien au moment de sa découverte, mais aussi à cause de l'étape d'avant.

Revenons à l'état de l'art en mathématiques. Euler a défini la fonction qui compte le nombre d'entiers positifs inférieurs (ou égaux) à n et qui sont relativement premiers avec n, dans les années 1750. Gauss lui a ensuite associé la lettre ϕ. On parle en français de la fonction indicatrice d'Euler. Les anglophones utilisent quant à eux le terme de

fonction totient d'Euler introduit par Sylvester en 1879. Parfois, l'on parle aussi de fonction « phi [1] » d'Euler. Or il est aisé de voir que $\#\mathsf{NonRelativementPremiers}(n) = n - 1 - \phi(n)$. Ce qui est très proche de $co\phi(n) = n - \phi(n)$. Cette dernière fonction a aussi déjà un nom mais plus récent : c'est la fonction « cototient » d'Euler, raison pour laquelle j'ai ajouté le préfix « co » devant ϕ. Vous allez me dire que tout est déjà connu alors. Presque ; par exemple, il est déjà connu que connaître le totient (ou le cototient) permet de casser RSA, car le totient est l'exposant privé du protocole RSA ; on sait même factoriser en temps polynomial tout nombre à partir du totient et donc à partir du cototient ou de $\#\mathsf{NonRelativementPremiers}$; donc mon intuition que factoriser correspond à une classe de complexité de comptage de solutions est pleinement démontrée. Néanmoins, j'ai été incapable de trouver une référence bibliographique antérieure faisant le lien entre ces résultats de théorie des nombres et la partie de la théorie de la complexité dévolue aux problèmes de comptage.

J'aurais pu définir différemment $\#\mathsf{NonRelativementPremiers}$ pour que cela soit égal à $co\phi(n) = n - \phi(n)$; il suffisait de réintégrer l'entier n ; mais du coup le problème de décision $\mathsf{NonRelativementPremiers}$ serait devenu trivial (toujours vrai comme pour les extensions linéaires d'un ordre).

Je trouve amusant que du point de vue de la factorisation et de la complexité algorithmique, il soit plus naturel de regarder la fonction cototient que totient. On a un résultat de complexité où le problème de comptage et de fournir une solution $\#\mathsf{NonRelativementPremiers}$ et $\mathsf{NonRelativementPremiers}$ ont exactement la même complexité à un facteur polynomial près. Si de plus, ces deux problèmes ne sont pas dans P, ce serait intéressant. En fait, on a déjà un résultat surprenant, puisque le problème de comptage est réductible au problème de fournir une solution explicite (mais pas au problème de décision qui est dans P, si $\#\mathsf{NonRelativementPremiers}$ n'est pas dans P). On a semble-t-il une hiérarchie du plus simple au plus difficile :

— Est-ce qu'il y a une solution ? : dans P,
— Combien y'a-t-il de solutions ? : complexité inconnue,
— Puis-je avoir une solution ? : complexité inconnue.

Pour les extensions linéaires d'un ordre et beaucoup d'autres problèmes, la hiérarchie est plutôt dans cet ordre :

— Est-ce qu'il y a une solution ? : dans P (temps constant),
— Puis-je avoir une solution ? : dans P,
— Combien y'a-t-il de solutions ? : $\#\mathsf{P}$-complet.

Du coup, même sans connaître les résultats de théorie des nombres factorisant grâce au totient, il était possible de suspecter que $\#\mathsf{NonRelativementPremiers}$ permet de factoriser pour que compter les solutions et fournir une solution puissent avoir la même complexité, pour que cette hiérarchie standard soit respectée.

À ma connaissance, c'est un problème ouvert de théorie de la complexité d'avoir automatiquement des réductions polynomiales entre ces 3 familles de problèmes : décision, solution, comptage, pour respecter cette hiérarchie standard. Bien sûr, on sait que :

— si on sait fournir une solution en cas d'existence, alors on sait aussi décider le problème,

1. J'ai écrit phonétiquement la lettre grecque, par analogie avec une recherche Google.

— si on sait compter les solutions, alors on sait aussi décider le problème.

Donc décider est tout en bas. Mais bizarrement, d'autres réductions sont triviales pour la plupart des problèmes NP-complets de base. Par exemple, si je sais décider l'existence d'une solution de 3-SAT ou si je sais les compter, alors il est trivial de tester les deux valeurs possibles d'une variable booléene, de simplifier les clauses restant à satisfaire, et donc de brancher sur la valeur booléenne qui donne un sous-problème ayant au moins une solution. Donc pour 3-SAT (et beaucoup d'autres problèmes), on a immédiatement décision équivalent à solution et comptage au moins aussi dur que décision et solution. Le problème de la factorisation des nombres entiers montre que :

— l'implication décider permet d'avoir une solution n'est pas triviale (et est peut-être même fausse),

— et que l'implication compter permet d'avoir une solution n'est pas triviale (bien que vérifiée par ce problème).

Pour finir cette section, je donne au lecteur qui n'aurait pas envie de chercher sur Internet, la preuve que compter c'est factoriser.

On peut avec la formule d'inclusion exclusion voir que $\#\mathsf{NonRelativementPremiers} = \#\mathsf{NRP}$ permet de factoriser tous les nombres entiers qui ont un facteur premier dont le carré est aussi un diviseur. Si $n = \prod_{i=1}^{j} p_i^{k_i}$ et au moins un des $k_i > 1$, alors $\#\mathsf{NRP}(n) = \sum_{\emptyset \subset I \subseteq [1,j]} (-1)^{|I|+1} \times (\prod_{i \in I} p_i^{k_i - 1} \times \prod_{i \in [1,j] \setminus I} p_i^{k_i} - 1)$. Donc $\#\mathsf{NRP}(n) = \prod_{i=1}^{j} p_i^{k_i - 1} \times poly(p_1, \ldots, p_j) + c$. Le terme constant $c = -1$ peut être déduit de la formule du binôme de Newton, avec $0 = (1-1)^j = \ldots$, car l'on exclut le sous-ensemble d'indice vide pour les valeurs de I. Le polynôme $poly(p_1, \ldots, p_j)$ importe peu. Le point à relever est juste que $\prod_{i \in I} p_i^{k_i - 1} > 1$ du fait de l'existence d'un facteur ayant une puissance supérieure à 1 dans la factorisation. Donc $1 < pgcd(\#\mathsf{NRP}(n)+1, n) < n$, puisque $\#\mathsf{NRP}(n) < n - 1$ et $\prod_{i \in I} p_i^{k_i - 1} | n$, $\prod_{i \in I} p_i^{k_i - 1} | \#\mathsf{NRP}(n) + 1$. Cette preuve est légèrement plus complexe que celle à base de (co)totient, à cause du terme constant, mais elle a le mérite que je l'ai trouvée sans aide extérieure et sans connaître les résultats existants.

Par contre, il reste le cas d'un nombre dont tous les facteurs sont distincts et en nombre au moins 3. Pour ce cas, j'ai trouvé la solution sur math.stackexchange.com [2]. La voici en français et j'ai aussi déterminisé au passage l'algorithme probabiliste fourni sur Internet.

— On commence par diviser par deux jusqu'à avoir un nombre impair à factoriser.

— Le totient $\phi(n)$ a la bonne idée d'être toujours pair sauf pour $n = 1$ et 2.
(Il est facile de le voir dans le cas des entiers impairs, car la formule d'inclusion exclusion pour le cototient va sommer $2^j - 1$ nombres impairs (produits de puissances de facteurs premiers impairs), donc le cototient sera impair, d'où le totient sera pair.)
Donc $\phi(n) = 2t$.

— On prend un nombre $1 < a < n$ au hasard :

  — si $pgcd(a, n) \neq 1$, on a trouvé un facteur, on continue avec un nombre plus petit ;

  — sinon, par le théorème d'Euler, $a^{\phi(n)} \equiv 1 \pmod{n}$, donc $a^{2t} \equiv 1 \pmod{n}$,

2. https://math.stackexchange.com/questions/191896/ https://math.stackexchange.com/questions/2916269/

$a^{2t} - 1 \equiv 0 \pmod{n}$; or $a^{2t} - 1 = (a^t + 1)(a^t - 1)$; donc $(a^t + 1)(a^t - 1) \equiv 0 \pmod{n}$. L'algorithme sur Internet se termine en constatant qu'avec « forte probabilité » (constante, nombre fini de tentatives), on aura $pgcd(a^t + 1, n)$ ou $pgcd(a^t - 1, n)$ qui fournit un diviseur non trivial. Comme tous les diviseurs de n sont dans le produit $(a^t + 1)(a^t - 1)$, dans le cas contraire, on a $a^t + 1 \equiv 0 \pmod{n}$ ou $a^t - 1 \equiv 0 \pmod{n}$.

Voilà la déterminisation de l'algorithme promise, j'ai du mal à croire qu'elle soit nouvelle bien que je l'ai trouvée par moi-même. Je suppose que les personnes sur Internet ont voulu laisser un peu de travail au lecteur, en donnant une forte indication avec cette histoire de « forte probabilité ». On va prendre deux valeurs de $1 < a < n$, $a = 2$ et $a = 3$. Supposons par l'absurde que ni 2 ni 3 ne sont des diviseurs de n, et que $pgcd(a^t + 1, n) \in \{1, n\}$ et $pgcd(a^t - 1, n) \in \{1, n\}$, pour $a = 2$ et $a = 3$, alors, par le théorème d'Euler et le raisonnement précédent, on a $a^t + 1 \equiv 0 \pmod{n}$ ou $a^t - 1 \equiv 0 \pmod{n}$, pour $a = 2$ et $a = 3$. (Tout le raisonnement marche avec deux valeurs $1 < a_1 < a_2 < n$ satisfaisant les conditions ci-dessus, mais d'un point de vue de la simplicité de la preuve et de l'efficacité algorithmique 2 et 3 suffisent.) On peut simplifier $a^t + 1 \equiv 0 \pmod{n}$ ou $a^t - 1 \equiv 0 \pmod{n}$ par l'existence d'un k_a tel que $|a^t - k_a n| = 1$. On a donc $|2^t - k_2 n| = 1$ et $|3^t - k_3 n| = 1$. Ce qui implique que $k_2 n - 1 \leq 2^t \leq k_2 n + 1$ et $k_3 n - 1 \leq 3^t \leq k_3 n + 1$. Par cette astuce, j'ai remplacé une disjonction de congruences par une conjonction d'inégalités et comme chacun sait l'explosion combinatoire vient des disjonctions ; c'est chouette la théorie des ordres :). Mais la bonne astuce finale c'est que $k_3 n - 1 \leq 3^t \leq k_3 n + 1$ implique que $(k_3 \times (\frac{2}{3})^t)n - (\frac{2}{3})^t \leq 2^t \leq (k_3 \times (\frac{2}{3})^t)n + (\frac{2}{3})^t$. Donc par transitivité des inégalités, on a $k_2 n - 1 \leq (k_3 \times (\frac{2}{3})^t)n + (\frac{2}{3})^t$ et $(k_3 \times (\frac{2}{3})^t)n - (\frac{2}{3})^t \leq k_2 n + 1$. Ce qui nous donne $(k_2 - k_3 \times (\frac{2}{3})^t)n \leq 1 + (\frac{2}{3})^t$ et $(k_3 \times (\frac{2}{3})^t - k_2)n \leq 1 + (\frac{2}{3})^t$. Si on prend les deux soustractions de ces deux inégalités, on obtient $(k_2 - k_3 \times (\frac{2}{3})^t)2n \leq 0$ et $(k_3 \times (\frac{2}{3})^t - k_2)2n \leq 0$. Donc $k_2 - k_3 \times (\frac{2}{3})^t = 0 \Leftrightarrow k_2 = k_3 \times (\frac{2}{3})^t \Leftrightarrow k_2 \times 3^t = k_3 \times 2^t$. Or on a aussi l'un de ces 4 cas :

— $n = \frac{k_2}{2^t+1} = \frac{k_3}{3^t+1}$,
— $n = \frac{k_2}{2^t+1} = \frac{k_3}{3^t-1}$,
— $n = \frac{k_2}{2^t-1} = \frac{k_3}{3^t+1}$,
— $n = \frac{k_2}{2^t-1} = \frac{k_3}{3^t-1}$.

D'où :

— $k_2 \times (3^t + 1) = k_3 \times (2^t + 1)$,
— $k_2 \times (3^t + 1) = k_3 \times (2^t - 1)$,
— $k_2 \times (3^t - 1) = k_3 \times (2^t + 1)$,
— $k_2 \times (3^t - 1) = k_3 \times (2^t - 1)$,

et donc :

— $k_2 = k_3$,
— $k_2 = -k_3$,
— $-k_2 = k_3$,
— $-k_2 = -k_3$.

k_2 et k_3 étant strictement positifs, les 2 cas au milieu sont impossibles. Mais le premier et le quatrième cas sont tout aussi impossibles, car $2^t + 1 \neq 3^t + 1$ et $2^t - 1 \neq 3^t - 1$. Ce raisonnement se généralisant par un simple « Rechercher-Remplacer » de « 2 » par

≪ a ≫ et de ≪ 3 ≫ par ≪ b ≫, on a en fait démontré qu'il y au plus un entier $1 < a < n$ tel que a est premier avec n, et que $pgcd(a^t + 1, n) \in \{1, n\}$ et $pgcd(a^t - 1, n) \in \{1, n\}$. Donc la ≪ forte probabilité ≫ annoncée sur Internet est de $1 - \frac{1}{n-2}$. C'est bien dommage qu'avec tous ces résultats préexistants, personne n'ait pensé aux problèmes de comptage pour ranger le problème de la factorisation.

2 Factoriser avec des graphes

Fin 2020, je lis le numéro 16 de ≪ 1024 ≫ le Bulletin de la Société informatique de France. Aux pages 121 à 132 se trouve un article de Jean Claude Derniame relatant ses travaux, sous la direction de Claude Pair, sur les cheminements dans les graphes (Derniame (2020)). Ces travaux ont été maintes fois redécouverts, voire plagiés, de manière partielle ou complète. J'y lis que les ensembles de chemins dans un graphe forment une structure de semi-anneau unitaire, donc je me dis ≪ Tiens, les nombres entiers positifs. ≫. Je m'aperçois rapidement que tout nombre entier peut être représenté par le nombre de chemins entre la source et le puit de graphes série-parallèle passablement triviaux. En effet, notons $x_i, i \geq 0$ un sommet associé à 2^i, on lui associe deux sommets intermédiaires h_i, b_i. Puis en mode petit-boutiste, on part de x_0 à gauche, suivi juste un peu à droite de h_0 en haut et de b_0 en bas, avec deux arcs $(x_0, h_0), (x_0, b_0)$, encore un peu plus à droite, on a x_1 avec deux arcs $(h_0, x_1), (b_0, x_1)$, etc. On voit tout de suite qu'il y a bien 2^i chemins distincts de x_0 à x_i. Donc pour représenter un entier positif n, on prend son écriture en base 2 (pour la construction donnée ici), on prend comme puit le dernier sommet $x_i, i \geq 0, 2^i \leq n$, et on ajoute un arc de $x_j, j \leq i$ à x_i, si le j-ème bit est à 1. La construction série-parallèle du graphe obtenu nécessite de partir de x_i et non de x_0, mais il n'y a rien de bien compliqué. Si je ne note que les nombres de chemins, plus le marquage s pour la source, p pour le puit, et a pour un arc allant direct vers le puit, l'entier 15 devient :

```
    1      2      4
1sa     2a     4a     15p
    1      2      4
```

Ces graphes sont suffisamment triviaux pour qu'il soit inutile de représenter les arcs avec un peu de pratique.

Mon père m'a dit une fois que Dieu immanent, présent partout et tout le temps, te répond quand tu t'adresses à lui. Il m'a raconté qu'une fois où il travaillait à son bureau à la maison de campagne, désespéré de voir ses idées d'auteur jeunesse plagiées, il s'était tourné vers Dieu pour crier sa détresse et que comme possédé, il était sorti de la maison, jusqu'au fossé du talus au bord du chemin, qu'il avait plongé la main parmi les pierres et trouvé un biface, c'est à dire un silex taillé de l'âge de pierre. Je ne sais toujours pas comment interpréter cette réponse de Dieu. Est-ce que Dieu voulait lui rappeler avec sympathie, voire ironie, d'où nous venons et donc le chemin parcouru par l'humanité depuis ces temps reculés, mais aussi le chemin qu'il reste à parcourir ? Est-ce que Dieu voulait lui dire de manière dure vous êtes des primitifs et vous allez retourner à ce stade-là prochainement ? Avec la conséquence que les problèmes de mon père n'étaient que du vent, un souffle temporaire, dans l'histoire des siècles ? Est-ce

que Dieu nous laisse le choix « Vois, je te propose aujourd'hui vie et bonheur, mort et malheur. » (Deutéronome chapitre 30 verset 15) ?

Quelques années plus tard, je prends ma voiture pour me rendre au point de départ d'une randonnée de fin d'année. Ne connaissant pas le trajet, je mets mon GPS mais celui-ci commence à m'afficher des trajets et des temps incohérents à mi-parcours. Je commence à paniquer et je crie à Dieu dans mon esprit, en pensant à ce que m'a dit mon père, mon GPS s'éteint, redémarre et se met à remarcher normalement.

Retour fin 2020, toujours la journée où je lis 1024 et je vois que je peux représenter tout entier par un graphe, excédé par les persécutions que je vis, le sabotage de mon travail avec la disparition de certains de mes devs ou de mes corrections de bug, je crie à Dieu en esprit, et je me souviens de l'algorithme de Shor, je vais dans la pièce où je stocke mes cours de fac, je prends le cours de calcul quantique et je commence à le feuilleter. Cela fait plus de 15 ans que je n'en ai pas fait, je le referme avec une seule idée : le produit tensoriel, la superposition d'états se traduit par un produit tensoriel. Du coup, dans les jours qui suivent, j'essaye de prendre la matrice d'un graphe série-parallèle et d'effectuer le produit tensoriel de cette matrice par elle-même, puis de calculer le pgcd entre le nombre que je veux factoriser et le nombre de chemins depuis le premier sommet, jusqu'au sommet courant, pour tous les sommets du graphe obtenu. Cela ne marche pas, je finis par trouver un algorithme qui marche mais dont je ne connais pas la complexité.

En théorie des ordres, il y a la ranked sum que je généralise en DAG-ranked sum (Directed Acyclic Graph). Soient $G_i, i \in I$ des DAGs, leur DAG-ranked sum c'est le graphe dont les sommets sont l'union disjointe des sommets des $G_i, i \in I$, chaque copie a les mêmes arcs, et en plus on ajoute un arc (x_i, y_j), si $x_i \in G_i, y_j \in G_j, i < j$ et le rang de x_i dans G_i est inférieur au rang de y_j dans G_j. Ainsi le rang ou niveau d'un sommet quelconque n'augmente pas, mais il y a plus d'éléments dans sa section initiale principale. Soit M_G la matrice triangulaire supérieure d'adjacence (TSA) d'un graphe orienté sans cycle. La matrice TSA de l'ordre total large à p éléments, c'est une matrice carrée $p \times p$ avec des 1 sur la diagonale et au dessus, et des 0 en dessous.

$$\begin{pmatrix} 1 & \dots & \dots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

Le produit tensoriel de cette matrice par M_G correspond à la matrice par bloc :

$$\begin{pmatrix} M_G & \dots & \dots & M_G \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & M_G \end{pmatrix}$$

Pour tout p, j'appelle la fonction qui associe cette matrice, ou son graphe associé, à la matrice M_G, ou son graphe associé G, la DAG-tensorial-p-sum. La DAG-ranked sum

de p copies de G, que j'appelle DAG-ranked-p-sum a pour matrice TSA quant à elle, la matrice :

$$\begin{pmatrix} M_G & L_G & \dots & L_G \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & L_G \\ 0 & \dots & 0 & M_G \end{pmatrix}$$

où la matrice L_G est obtenue en ajoutant les arcs allant de tout sommet initial de rang/niveau donné vers tout sommet terminal de rang/niveau supérieur.

Si l'on s'intéresse aux pistes (chemins orientés) qui partent du premier sommet, dans la DAG-tensorial-p-sum, on va repartir du graphe série-parallèle de tout à l'heure et on va le dupliquer, par exemple de haut en bas.

```
    1     2     4
1s     2     4     15
    1     2     4
    1     6     20
0      4     16    75
    1     6     20
```

Seul le 75 dans cet exemple dépend du nombre 15. Tout le reste est un motif commun à tous les nombres. On peut constater que le pgcd de 6 et de 15 est 3 ce qui permet de factoriser 15. Je suis incapable de quantifier l'apport du point de vue de l'algorithme de factorisation du 75, c'est-à-dire la partie variable. Mais l'on peut voir aisément que la partie commune tombe toujours sur un diviseur du nombre par calcul de pgcd. Il faut un exemple plus volumineux pour le voir.

```
    1     2     4     8
1s     2     4     8     16
    1     2     4     8
    1     6     20    36
0      4     16    28    88
    1     6     20    36
    1
0      6 ...
    1
    1
0      8 ...
    1
    1
0      10 ...
    1
...
```

Et oui, c'est complètement con. On a tous les nombres pairs qui apparaissent bien sagement. Donc forcément, on va finir par tomber sur un diviseur. C'est bien un algorithme et non une heuristique, mais bien malin la personne qui trouvera sa complexité dans le

pire cas XD. En fait, je pense que l'on peut déguiser l'algorithme de factorisation que je résumerai par « boucle for + pgcd » de bien des manières.

Néanmoins, il y a toujours la possibilité que la partie variable apporte quelque chose. De plus, j'ai un autre « théorème publicitaire ». Comment ça un théorème publicitaire, les mathématiques sont une discipline sérieuse vont répondre les gens coincés. Et bien oui, je ne suis sans doute pas le premier à écrire un théorème publicitaire, mais sans doute le premier à écrire cette formulation, non sans humour.

Théorème 2.1. *S'il existe un algorithme de factorisation en temps $O(f(n))$, il en existe un qui construit un graphe série-parallèle, dont le nombre de chemins entre la source et le puit est non relativement premier avec n en temps $O(f(n) + Poly(\lg(n)))$.*

Donc si c'est polynomial, cela l'est aussi avec un algorithme à base de graphes, au moins en « trichant », c'est-à-dire en commençant par un vrai algorithme. Quand j'étais en thèse, j'avais dit à Emmanuelle Lebhar que je m'étais demandé pourquoi l'évolution avant tant favorisé la connerie, et que je m'étais dit que l'une des raisons était peut-être que le graphe des idées intelligentes n'est pas petit-monde alors qu'une fois plongé dans celui plus large où on inclut les idées connes, cela devient petit-monde.

Du coup, quelles sont les idées intelligentes ?

— L'algorithme ? Bof, à tester mais peu probable.

— Étudier des opérations arithmétiques sur les graphes ? L'addition c'est facile. Mais par exemple, comment implémenter un algorithme de multiplication rapide avec des graphes ? Pas forcément très utile, à première vue cela ressemble plus à la variante du marathon avec le sac à dos sur les épaules, mais si l'on n'essaie pas, on est certain de ne rien trouver d'intéressant. Il y a tout de même un truc sympa, la somme c'est la composition parallèle et le produit c'est la composition série, si l'on ne se préoccupe que du nombre de chemins. Le vrai problème se trouve sur la « normalisation » du graphe série-parallèle pour qu'il ressemble aux graphes ayant une structure régulière comme avant : degré entrant au plus deux sauf pour le puit, degré sortant au plus 3, pas d'arc qui « saute » un niveau sauf pour ceux qui vont directement vers le puit. Cette normalisation est triviale à faire dans le cas de la composition parallèle de deux graphes normalisés. Si l'on sait normaliser, on peut ensuite reconvertir dans une notation binaire classique en temps linéaire. Il suffirait de savoir normaliser ou bien reconvertir directement dans une notation binaire classique en temps quasi linéaire pour avoir un algorithme de multiplication rapide.

— Effectuer des calculs avec un langage « haut niveau » d'opérations matricielles comme la DAG-tensorial-p-sum ou la DAG-ranked-p-sum en variantes de boucles for ? Là oui, on a déjà un précédent avec le théorème de Ben-Or and Cleve (1988) qui permet de calculer les formules arithmétiques avec 3 registres, et un relecteur anonyme de Flarup and Lyaudet (2009) nous avait fait remarquer que cela se traduit par itérer des produits de matrice 3×3. Beaucoup d'autres résultats existent sur le produit itéré de matrices, voir par exemple Immerman and Landau (1995). Il y aussi le théorème de Savitch montrant que $\mathsf{PSpace} = \mathsf{NPSpace}$, avec sa preuve à base de graphes. Des modèles de calcul à base de graphes ou de matrices, cela vaut le coup de regarder, à mon avis.

Si cela se trouve aucune de ces idées ne valent quelque chose et si cela se trouve toutes ont un intérêt. Est-ce que c'est un biface ou bien mieux que Dieu m'a donné ou que j'ai trouvé en cherchant Dieu ? Ce dont je suis sûr c'est que les idées de la section précédente qu'il m'a données juste avant Pâques sont plus évoluées et moins spéculatives à mes yeux, que celles que j'ai cherchées par mes propres moyens, même en l'invoquant. Si j'avais inversé ces deux sections, les lecteurs auraient sans doute arrêté de lire avant les meilleures idées.

J'ai présenté les idées de ces deux premières sections à l'ENS Lyon le 3 juin 2022, mais pas aux mêmes personnes et de manière beaucoup plus rapide. J'ai rédigé la première section dans la soirée du 5 juin. Puis j'ai fait une pause d'un mois pour améliorer mon article « Diviser n'est pas régner ? », notamment en implémentant tous les algorithmes en PHP. Un truc amusant c'est que le matin du 6 juin alors que je somnolais dans mon lit, j'ai entendu plusieurs fois une voix pleine de joie dire « 5 millions dollars ». Et ce n'était pas ma voix, donc si à tout hasard, cela correspond vraiment à une personne réelle, j'ai juste envie de dire : « Zachée, tu t'es trompé d'arbre et de personne, mais tu peux quand même donner la moitié aux pauvres ! Ahahahahah ! »(Prends le temps de lire Luc chapitre 19 versets 1 à 10.).

3 Fibolous

Pour finir, voici une idée que j'ai eu en 2012 ou 2013. Quand je me suis rendu à la conférence CSR 2008 pour présenter notre article avec Uffe sur la complexité de Valiant, ce cadre de travail a fortement surpris Leonid Levin car l'on se permettait des constantes arbitraires pour calculer nos polynômes. Et il m'a dit qu'avec des calculs sur des nombres entiers de taille arbitraire, on pouvait factoriser en temps polynomial. Je ne suis absolument pas un spécialiste de théorie des nombres et je n'ai même pas fait de recherche bibliographique pour comprendre exactement ce qu'il voulait dire. Je ne sais si sa remarque est un résultat bien connu du domaine, ou si, comme avec certains passages de ses articles, même ses élèves sont venus lui réclamer de redescendre un peu à leur niveau et de donner plus d'explications. Mais du coup avec mon niveau en mathématiques qui est assez bon mais clairement pas du tout au niveau de quelqu'un comme lui, j'ai commencé à jouer avec l'idée de générer des grands nombres puis de faire un calcul de pgcd pour essayer de factoriser. Du coup, les puissances de 2 ce n'est pas très efficace, le candidat suivant qui m'est venu à l'esprit de suite à croissance exponentielle, c'est la suite de Fibonacci. Alors techniquement, on n'est pas obligé de laisser croître la suite sans borner les nombres, on peut tout calculer modulo le nombre que l'on cherche à factoriser. Et du coup, on obtient une heuristique où on contrôle le temps de calcul que l'on est prêt à y mettre, qu'il soit linéaire, quadratique ou autre. Cela m'a permis de voir que certains nombres premiers n'apparaissent jamais dans la suite de Fibonacci. Par exemple, $4181 = 37 \times 113$ est le plus petit nombre non factorisable par cette heuristique, elle boucle indéfiniment sur les mêmes nombres modulo 4181. Il est donc facile de vérifier si un nombre premier apparaît dans la suite de Fibonacci, il suffit de le multiplier par 37 qui est le plus petit nombre premier qui n'apparaît jamais, et de regarder si la suite de nombres de l'heuristique boucle. Cette idée loufoque contient déjà en germe l'idée d'étudier les nombres non relativement

premiers pour factoriser, l'étape la plus intéressante étant venu de l'idée de les compter par laquelle j'ai commencé cet article.

4 Conclusion

Mon premier directeur de thèse Vincent Bouchitté était aussi mon professeur de cryptographie pendant ma maîtrise d'informatique à l'Université Claude Bernard Lyon 1. Je me souviens que quand il nous a présenté RSA, il nous a dit qu'il avait entendu plusieurs fois des affirmations comme quoi casser RSA était NP-dur, mais qu'il n'en avait jamais vu la preuve, avant d'avoir un sourire dubitatif comme cela lui arrivait parfois. Comme j'étais déjà paranoïaque, je me suis dit « Pourquoi cette publicité mensongère ? ». Est-ce que certaines personnes savent déjà casser RSA mais ne souhaitent pas que l'on change de cryptosystème à clé publique ? (En pratique, c'est vrai dans certains endroits sur les communications des particuliers et des entreprises, puisque certains États imposent une limite de taille sur les clés utilisées.) En tout cas, depuis ce jour, j'en ai gardé l'idée que RSA voire factoriser n'était sans doute pas si difficile. Pendant ma thèse, j'avais constaté qu'il y avait une sorte de « meta-théorème » : « Deux c'est facile, trois c'est difficile. ». Par exemple, les graphes de degré maximum 2 comparés aux graphes de degré maximum 3 et ses nombreuses conséquences comme le nombre de réunion de parties dans mon article « Diviser n'est pas régner ? », la programmation linéaire ou semi-définie avec les formes quadratiques comparées aux polynômes de degré supérieur, etc. Cela ne marche pas pour tout mais cela marche souvent. Si quelqu'un trouve un formalisme satisfaisant pour ce « meta-théorème », je serais très heureux de l'apprendre. Mon opinion courante, appuyée par ce « meta-théorème » et le fait que compter permet de factoriser, beaucoup plus simplement (en termes de preuve mathématique) dans le cas où il n'y a que deux nombres premiers diviseurs, est qu'il est possible que factoriser soit dur mais seulement s'il y a au moins trois nombres premiers diviseurs, et que le cas avec deux nombres premiers diviseurs pour RSA soit polynomial.

Merci Dieu ! Merci Père ! Merci Seigneur ! Merci Saint Esprit ! Merci Jean Claude Derniame ! Merci Claude Pair ! Merci Leonid Levin ! Merci Vincent Bouchitté ! Je remercie aussi Frédéric Mazoit de m'avoir à peu près dit : « Hé ! Gros malin qui fuit la biblio et cogite tout seul, regarde du côté de l'indicatrice d'Euler ! ».

Références

Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, 6-8 May 1991, New Orleans, Louisiana, USA, 1991. ACM.

M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160 : 781–793, 2004.

M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. In *STOC 1988, Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 254–257. ACM, 1988.

G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *STOC* DBL (1991), pages 175–181.

J. C. Derniame. À propos du cheminement dans les graphes. *1024 Bulletin de la SiF*, 16, 2020.

U. Flarup and L. Lyaudet. On the expressive power of permanents and perfect matchings of matrices of bounded pathwidth/cliquewidth (extended abstract). In *CSR 2008, Computer Science in Russia*, volume 5010 of *Lecture Notes in Computer Science*, pages 180–193. Springer Verlag, 2008.

U. Flarup and L. Lyaudet. On the expressive power of permanents and perfect matchings of matrices of bounded pathwidth/cliquewidth. *Theory of Computing Systems*, 2009.

N. Immerman and S. Landau. The complexity of iterated multiplication. *Information and Computation*, 116(11) :103–116, 1995.

P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5) :1484–1509, oct 1997. doi : 10.1137/s0097539795293172. URL `https://doi.org/10.1137/s0097539795293172`.