Nouveaux résultats expérimentaux concernant la conjecture de Goldbach

J-M. Deshouillers
Mathématiques Stochastiques, Université Victor Segalen Bordeaux 2
F-33076 Bordeaux Cedex, France
J-M.Deshouillers@u-bordeaux2.fr

H.J.J. te Riele
CWI, P.O. Box 94079, 1090 GB Amsterdam, Pays-Bas
Herman.te.Riele@cwi.nl

Y. Saouter
Institut de Recherche en Informatique de Toulouse
118 route de Narbonne, F-31062 Toulouse Cedex, France
Yannick.Saouter@irit.fr

31 mars 1998

Résumé La conjecture de Goldbach énonce que tout entier pair ≥ 4 peut être écrit comme une somme de deux nombres premiers. On sait qu'elle est vraie jusqu'à 4×10^{11}. Dans cet article, de nouvelles expérimentations menées sur un super-ordinateur Cray C916 et sur un serveur de calcul SGI à 18 R8000 CPU sont décrites, qui étendent cette limite à 10^{14}. Deux conséquences sont que (1) sous l'hypothèse de Riemann généralisée, tout nombre impair ≥ 7 peut s'écrire comme une somme de trois nombres premiers, et (2) en supposant vraie l'hypothèse de Riemann, tout entier positif pair peut s'écrire comme une somme d'au plus quatre nombres premiers. De plus, nous avons vérifié la conjecture de Goldbach pour tous les nombres pairs dans les intervalles $[10^{5i}, 10^{5i} + 10^8]$, pour $i = 3, 4, \ldots, 20$ et $[10^{10i}, 10^{10i} + 10^9]$, pour $i = 20, 21, \ldots, 30$.
Un modèle heuristique est fourni qui prédit le nombre moyen d'étapes nécessaires pour vérifier la conjecture de Goldbach sur un intervalle donné. Nos résultats expérimentaux sont en bon accord avec cette prédiction. Cela ajoute à l'évidence de la vérité de la conjecture de Goldbach.

Remerciements. Le premier auteur a bénéficié du support du CNRS et des universités Bordeaux 1 et Bordeaux 2. Le second auteur remercie Walter Lioen pour son aide à prouver la primalité de nombreux grands nombres avec les programmes de Cohen, Lenstra et Winter, et de Bosma et Van der Hulst. L'accès à l'ordinateur de calcul vectoriel Cray C916 au Centre de Calcul académique d'Amsterdam (SARA) a été possible grâce à la Fondation nationale allemande d'aide au calcul (NCF). L'accès au serveur de calcul Power Challenge Array R10000 a été possible grâce au Centre Charles Hermite de Nancy, et à l'INRIA de Lorraine.

Modélisation, analyse et simulation (MAS), Rapport MAS-R9804, 31 mars 1998.

1991 Classification des sujets mathématiques : Primaire 11P32. Secondaire 11Y99.
1991 Système de classification des résultats de calculs : F.2.1.
Mots et phrases clefs : conjecture de Goldbach, somme de nombres premiers, test de primalité, calcul vectoriel, Cray C916, cluster de stations de travail.
Note : Cet article apparaîtra dans les Proceedings de ANTS-III (Symposium de théorie des nombres algorithmique III, Reed College, Portland, Oregon, USA, 21-25 juin 1998).
La contribution du second auteur a été financée par le projet MAS2.5 "Théorie des nombres computationnelle et sécurité des données".

1. Introduction

La conjecture *binaire* de Goldbach (BGC) énonce que tout nombre entier pair ≥ 4 peut être exprimé comme une somme de deux nombres premiers. Par G_2 nous dénotons la plus petite borne supérieure pour le nombre G avec la propriété que tous les nombres pairs n tels que $4 \leq n \leq G$ peuvent s'écrire comme somme de deux nombres premiers. On sait que $G_2 \geq 4 \times 10^{11}$ [15, 17, 7, 16].
La conjecture de Goldbach *ternaire* (TGC) énonce que tout nombre entier impair ≥ 7 peut être exprimé comme la somme de trois nombres premiers. De façon évidente, la vérité de BGC implique la vérité de TGC.
En 1923, Hardy et Littlewood [8] ont prouvé que, sous l'hypothèse d'une faible version de l'hypothèse de Riemann généralisée (GRH), il existe un entier positif M_0 tel que TGC est vérifiée par tous les nombres entiers impairs $\geq M_0$. En 1937, Vinogradov [18] a prouvé, inconditionnellement, qu'il existe un entier positif N_0 tel que TGC est vérifiée pour tous les nombres entiers impairs $\geq N_0$.
En 1989, Chen et Wang [3] ont montré qu'on peut prendre $N_0 = 10^{43000}$, et en 1993 [4] ils ont montré, en supposant GRH, que l'on peut prendre $M_0 = 10^{50}$. Très récemment, Zinoviev [19] a prouvé, en supposant GRH, qu'on peut prendre $M_0 = 10^{20}$. En utilisant des calculs classiques, selon Schoenfeld [14], ce résultat implique [6] le

Théorème A *Si GRH est vraie et si $G_2 \geq 1.615 \times 10^{12}$, alors tout nombre entier impair ≥ 7 peut être exprimé comme la somme de trois nombres premiers.*

C'est une de nos motivations pour la présente étude.

Remarque Dans [13], le troisième auteur a prouvé, inconditionnellement, la vérité de TGC jusqu'à 10^{20} en calculant une séquence croissante d'environ 2.5×10^8 nombres premiers $q_0, q_1, \ldots\ldots, q_Q$ telle que $q_0 < 4 \times 10^{11}$, $q_{i+1} - q_i < 4 \times 10^{11}$ pour tout $0 \leq i \leq Q-1$ et $q_Q > 10^{20}$. Cela montre que près de tout nombre impair $N < 10^{20}$, il y a un nombre premier q tel que $N - q < 4 \times 10^{11}$ et par [16] $N - q$ peut être exprimé comme une somme de deux nombres premiers.

Une seconde motivation était le résultat suivant de Kaniecki [10] :

Théorème B *Si l'hypothèse de Riemann (RH) est vraie et si $G_2 \geq 1.405 \times 10^{12}$, alors tout nombre entier positif pair peut être exprimé comme la somme d'au plus quatre nombres premiers.*

Sans aucune hypothèse, Ramaré [12] a prouvé que tout nombre entier pair positif est la somme d'au plus six nombres premiers.
Dans ce papier, nous rapportons les résultats d'expériences extensives dont l'effet est le
Théorème 1 *On a $G_2 \geq 10^{14}$,*
de telle façon que les suppositions sur G_2 dans les Théorèmes A et B sont satisfaites.
De plus, nous avons vérifié que tous les nombres pairs dans quelques intervalles donnés sont sommes de deux nombres premiers, notamment :

Théorème 2 *Tous les nombres entiers pairs dans les intervalles* $[10^{5i}; 10^{5i} + 10^8]$*, pour* $i = 3, 4, \ldots, 20$ *et* $[10^{10i}; 10^{10i} + 10^9]$*, pour* $i = 20, 21, \ldots, 30$ *sont sommes de deux nombres premiers.*

Nous avons vérifié BGC avec un algorithme qui a été utilisé, mais non fourni explicitement par Mok-Kong Shen [15]. De plus pour étendre l'intervalle sur lequel BGC est connue comme étant vraie d'un facteur de 250, nous donnons un modèle heuristique qui prédit le nombre moyen d'étapes nécessaires pour vérifier BGC avec cet algorithme. Cela ajoute une évidence théorique à l'évidence numérique déjà accablante de la vérité de BGC.

2. Deux algorithmes pour vérifier la conjecture de Goldbach binaire sur $[a, b]$

Les algorithmes connus pour vérifier la conjecture de Goldbach sur un intervalle donné $[a, b]$ consistent à trouver deux ensembles de nombres premiers $\mathcal{P}$ et $\mathcal{Q}$ tels que $\mathcal{P} + Q$ couvre tous les nombres pairs dans $[a, b]$.

Appelons p_i le i-ème nombre premier impair. Une approche, telle qu'appliquée en [17, 7, 16], consiste à trouver, pour tout nombre pair $e \in [a, b]$, le plus petit nombre premier impair p_i tel que $e - p_i$ est un nombre premier. Cela revient à prendre pour $\mathcal{P}$ les nombres premiers impairs $p_1, p_2, \ldots, p_m$ pour m adéquat et à prendre

$$\mathcal{Q} = \mathcal{Q}(a, b) = \{q \mid q \text{ premier et } a - \epsilon_a \leq q \leq b\}$$

pour un ϵ_a convenablement choisi. Une série d'ensembles de nombres pairs $\mathcal{E}_0 \subset \mathcal{E}_1 \subseteq \mathcal{E}_2 \subseteq \ldots$ est alors générée, définie par $\mathcal{E}_0 = \varnothing$,

$$\mathcal{E}_{i+1} = \mathcal{E}_i \cup (\mathcal{Q}(a, b) + p_{i+1}), \qquad i = 0, 1, \ldots, \text{ [1]}$$

jusqu'à ce que pour un certain j l'ensemble $\mathcal{E}_j$ couvre *tous* les nombres pairs dans l'intervalle $[a, b]$. L'ensemble $\mathcal{Q}(a, b)$ est engendré avec le crible d'Eratosthène : c'est la partie du calcul la plus chronophage. Pour le choix de ϵ_a, il est suffisant que ϵ_a soit plus grand que le plus grand nombre premier impair p_j utilisé pour engendrer les ensembles $\mathcal{E}_j$. Cette approche permet de fournir, pour tout nombre entier pair $e \in [a, b]$, le plus petit nombre premier p tel que $e - p$ est premier (la paire $(p, e - p)$ est alors appelée la décomposition de Goldbach minimale de e). Dans les calculs utilisés pour vérifier la conjecture de Goldbach jusqu'à 4×10^{11} [16], le plus grand *petit* nombre premier impair nécessaire a été $p_{446} = 3163$ (c'est le plus petit nombre premier p pour lequel $244\,885\,595\,672 - p$ est un nombre premier). Une partie coûteuse de cette approche est que quasiment tous les nombres premiers de l'intervalle $[a, b]$ doivent être déterminés.

Une approche plus efficace, comme celle appliquée dans [15], est de trouver, pour tout nombre pair $e \in [a, b]$, un nombre premier q, proche de a, pour lequel $e - q$ est un nombre premier. Cela revient à choisir pour $\mathcal{P}$ l'ensemble de tous les nombres premiers impairs environ jusqu'à $b - a$ et pour $\mathcal{Q}$ les k plus grands nombres premiers $q_1 < q_2 \ldots < q_k$ en-dessous de a, pour k adéquat. Pour la vérification effective de l'intervalle $[a, b]$, on génère l'ensemble des nombres pairs $\mathcal{F}_0 \subset \mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$, défini par $\mathcal{F}_0 = \varnothing$,

$$\mathcal{F}_{i+1} = \mathcal{F}_i \cup (\mathcal{P} + q_{i+1}), \qquad i = 0, 1, \ldots,$$

jusqu'à ce que pour un certain j l'ensemble F_j couvre *tous* les nombres pairs dans l'intervalle $[a, b]$. Le grand ensemble $\mathcal{P}$ est engendré avec le crible d'Eratosthène, mais le travail n'a à être fait qu'une seule fois si nous fixons la longueur $b - a$ des intervalles $[a, b]$. Les nombres premiers dans $\mathcal{Q}$ dépendent de a et pourraient aussi

1. Par $\mathcal{Q}(a, b) + p_{i+1}$, nous voulons dire, comme d'habitude, l'ensemble $\{q + p_{i+1} | q \in \mathcal{Q}(a, b)\}$.

être générés par le crible d'Eratosthène. Pourtant, puisque nous n'avons besoin que de quelques centaines de tels nombres premiers et puisqu'ils n'excèdent pas 10^{14}, il est plus économique d'utiliser les résultats de Jaeschke [9] par lesquels pour tout nombre premier nous avons seulement besoin de faire quelques tests de pseudo-primalité, aussi longtemps qu'ils n'excèdent pas 3.4×10^{14}. Un inconvénient de cette approche est qu'en général, elle *ne trouve pas*, la décomposition de Goldbach *minimale*.
Dans cette étude, nous avons choisi d'implémenter la seconde approche. En plus d'étendre G_2 autant qu'il est possible, nous sommes intéressés par le *nombre d'étapes nécessaires* dans les algorithmes ci-dessus, pour vérifier BGC. Dans la prochaine section, nous discutons d'un modèle heuristique qui est capable de prédire le nombre d'étapes *moyen* précisément.

3. Prédire le nombre moyen d'étapes nécessaires pour vérifier BGC sur $[a, b]$

Nous présentons quelques heuristiques pour estimer le nombre moyen d'étapes nécessaires pour générer les ensembles $\mathcal{F}_i, i = 0, 1, \ldots$ jusqu'à ce que tous les nombres pairs dans $[a, b]$ soient couverts.
Prenons $l = b - a$ assez grand, comparé avec a, de telle façon que nous puissions trouver suffisamment de nombres premiers q dans le voisinage de a pour répondre à notre objectif. Le nombre de nombres premiers dans $\mathcal{P}$ est approximativement $\pi(l)$. Pour chaque nombre premier $q \in \mathcal{Q}$, l'ensemble $\mathcal{P}+q$ couvre environ $\pi(l)$ éléments dans $[a, b]$, i.e. une proportion d'environ $1-2\pi(l)/l$ des nombres pairs dans $[a, b]$ n'est pas couverte. Si l'on suppose, ce qui n'est pas le cas, une indépendance statistique entre le fait d'être couvert par $\mathcal{P} + q$ et le fait d'être couvert par $\mathcal{P} + q'$ ainsi qu'une hypothèse supplémentaire d'uniformité, on peut s'attendre à ce que, en moyenne, tous les nombres pairs soient couverts à l'aide de k éléments q quand $(1 - 2\pi(l)/l)^k$ est grosso-modo égal à $2/l$, l'inverse du nombre de nombres pairs dans $[a, b]$. Si $l = 10^8$, cela amène à $k \approx 145$ et pour $l = 10^9$ cela entraîne $k \approx 187$. Une étude plus détaillée du modèle probabiliste amène à un comportement de Poisson pour le nombre d'entiers qui ne sont pas couverts ; dans ce modèle, pour $k \approx 148$ dans le cas où $l = 10^8$ (et $k \approx 191$ quand $l = 10^9$) la probabilité de couvrir l'intégralité de l'intervalle $[a, b]$ est proche de $1/2$. Pourtant, cela n'est pas en accord avec nos observations expérimentales décrites dans les sections suivantes. Bien qu'une sorte de quasi-indépendance statistique semble une hypothèse naturelle, la distribution uniforme des nombres premiers n'est définitivement pas une hypothèse décente.
Un premier défaut d'uniformité vient de la raréfaction des nombres premiers (la densité locale des nombres premiers autour de x décroît lorsque x croît). Considérer seulement les grands nombres premiers, par exemple, ceux entre 10^7 et 10^8 pour couvrir un intervalle de longueur 9×10^7, amène à la valeur $k \approx 150$; ceci est en accord avec la valeur moyenne expérimentale des k observés (cf. Section 5.1).
Un second et plus important manque d'uniformité est de nature arithmétique. Choisissons un petit nombre premier r et considérons les décompositions de Goldbach de tous les nombres pairs qui sont premiers à $R = 3.5 \ldots r$. Pour chaque grand nombre q (premier, et donc premier à R), tous les nombres premiers $p \in P$ qui vérifient $(p + q, R) > 1$ ne peuvent être utilisés pour décomposer nos nombres. Le nombre de classes admissibles de premiers est ainsi $(3-2)(5-2)\ldots(r-2)$ et la proportion de nombres premiers utiles dans $\mathcal{P}$ est donc $\frac{(3-2)(5-2)\ldots(r-2)}{(3-1)(5-1)\ldots(r-1)}$. Ainsi, pour chaque nombre premier q, l'ensemble $\mathcal{P} + q$ contient environ $\frac{(3-2)(5-2)\ldots(r-2)}{(3-1)(5-1)\ldots(r-1)}\pi(l)$ nombres pairs différents et alors, la proportion de nos nombres pairs dans $[a, b]$ qui sont couverts en une étape est

$$\frac{(3-2)(5-2)\ldots(r-2)}{(3-1)(5-1)\ldots(r-1)}\pi(l)/\left(\frac{(3-1)(5-1)\ldots(r-1)}{3.5\ldots r}\frac{l}{2}\right)$$

i.e.,

$$2 \prod_{\substack{3 \le s \le r \\ s \text{ premier}}} \left(1 - \frac{1}{(s-1)^2}\right) \frac{\pi(l)}{l} = C(r) \frac{\pi(l)}{l}.$$

Par le même raisonnement que ci-dessus, nous nous attendons à ce que k soit proche de la solution de $\left(1 - C(r)\frac{\pi(l)}{l}\right)^k = \frac{2R}{\phi(R)l}$. Pour $r = 97$ et $l = 10^8$, cela amène à $k \approx 206$ et pour $l = 10^9$ nous trouvons $k \approx 270$.

Cela s'accorde bien aux résultats de nos expérimentations et cela implique, comme on peut s'y attendre, que pour les nombres pairs dans $[a, b]$ qui ne sont pas premiers à $R = 3.5 \dots r$, il est en général plus facile de trouver une décomposition de Goldbach que pour ceux qui sont premiers à R. À nouveau, si nous améliorons ce modèle par des considérations probabilistes de Poisson, et en considérant la raréfaction des nombres premiers, nous sommes amenés à $k \approx 214$ quand $l = 10^8$, qui est, ici aussi, en bon accord avec les données expérimentales de la Section 5.1. Ce raisonnement probabiliste sera développé dans un article à venir.

4. Calculs qui étendent G_2 de 4×10^{11} à 10^{14}

Nous avons adopté l'approche de Shen, décrite dans la Section 2, pour vérifier la conjecture de Goldbach aussi loin que possible au-delà de la borne connue de 4×10^{11}.

Les intervalles $[a, b]$ ont été choisis de façon à être de longueur 10^8 ou 128×10^6 ou 10^9. Le plus grand nombre premier dont on a besoin dans l'ensemble $\mathcal{P}$ est proche de $b - q_1$. Par le théorème des nombres premiers, $q_1 \approx a - k \log a$, de telle façon que $b - q_1 \approx b - a + k \log a$. Comme valeurs *maximum* de k, nous avons trouvé dans nos expériences que $k = 430$ était une valeur suffisante. Pour $a \approx 10^{14}$, cela implique que le plus grand nombre premier dans l'ensemble $\mathcal{P}$ doive avoir une taille d'au moins $10^9 + 1.4 \times 10^4$ pour $b - a = 10^9$. Dans notre implémentation effective, nous avons choisi que $\mathcal{P}$ contienne les nombres premiers impairs jusqu'à $10^8 + 10^5$ dans le cas $b - a = 10^8$, et ceux jusqu'à $10^9 + 10^6$ dans le cas $b - a = 10^9$.

Pour la génération effective des nombres premiers proches de a, nous avons utilisé les résultats des calculs de Jaeschke [9], qui énoncent que si un entier positif $n < 215\,230\,289,8747$ [2] est un fort pseudo-premier par rapport aux cinq premiers nombres premiers $2, 3, 5, 7, 11$, alors n est premier ; les bornes correspondantes pour les six premiers et sept premiers nombres premiers sont 3 474 749 660 383 et 341 550 071 728 321, respectivement.

Initialement, à la fois le second et le troisième auteurs ont vérifié la BGC jusqu'à 10^{13}, indépendamment, sur un calculateur vectoriel Cray C916 resp. sur un serveur de calcul SGI avec 18 R8000 CPU. Après avoir appris les résultats l'un de l'autre, ils ont décidé de travailler ensemble pour atteindre la borne 10^{14}. Le second auteur a vérifié la BGC sur les intervalles $x \times 10^{13}$ pour $x = [2, 4], [6, 8], [9, 10]$ et le troisième auteur ceux pour les cas $x = [1, 2], [4, 6], [8, 9]$.

4.1 *Expérimentations sur le calculateur vectoriel Cray C916*

Nous avons implémenté l'algorithme de Shen sur un ordinateur vectoriel Cray C916 comme suit.

Au grand ensemble de nombres premiers impairs $\mathcal{P}$, nous associons un tableau d'entiers longs (long-bit) appelé ODD, dans lequel chaque bit représente un nombre impair $< 10^9 + 10^6$, le bit valant 1 si le nombre impair correspondant est premier, et 0 s'il est composé. À $\mathcal{F}_i$, nous associons un tableau de bits similaire

2. ?

appelé SIEVE, ayant la même longueur que ODD. Le premier bit de SIEVE représente le nombre impair $q_1 + 3$, le second bit $q_1 + 5$, et, et en général, le bit i représente le nombre pair $q_1 + 2i + 1$. Initialement, ODD est copié dans SIEVE, rendant le bit i du tableau SIEVE égal à 1 si $2i + 1$ est premier, indiquant que $q_1 + 2i + 1$ peut s'écrire comme une somme de deux nombres premiers q_1 et $2i + 1$. Maintenant le tableau SIEVE représente l'ensemble $\mathcal{F}_1$. Dans un second temps, le tableau SIEVE est "ou"-é avec une version décalée à droite du tableau ODD, où le décalage est égal à $(q_2 - q_1)/2$. Il est facile de voir que maintenant, le tableau SIEVE représente l'ensemble $\mathcal{F}_2 = \mathcal{F}_\infty \cup (\mathcal{P} + q_2)$. En général, $\mathcal{F}_{i+1}$ est engendré à partir de $\mathcal{F}_i$ en faisant une opération "ou" entre le tableau SIEVE et le tableau ODD, décalé à droite d'un décalage de $(q_{i+1} - q_1)/2$.

Bien sûr, ces étapes peuvent être effectuées très efficacement sur un Cray C916. Nous avons compressé 64 bits en un mot et vectorialisé les opérations "ou". La vérification pour savoir si *tous* les bits du tableau SIEVE sont devenus des 1 n'est effectuée que lorsque la chance d'occurrence de cet événement est devenue suffisamment grande (après 170 étapes, dans notre programme). Dès que le nombre de bits 0 est tombé en-dessous de 4, les nombres restant "obstinément" pairs sont listés de manière à "voir" une sortie intermédiaire.

En une passe typique, on traite 1000 intervalles consécutifs de longueur 10^9. Proche de 10^{14}, le temps pour générer 1000×430 grands nombres premiers était environ 5000 secondes CPU, et le temps de crible total était environ de 13 200 secondes. Le nombre moyen d'étapes (sur 1000 intervalles consécutifs) à chaque passe varie entre 269 et 271 avec une déviation standard entre 18 et 20. Le temps total CPU (en ayant une priorité faible) utilisé pour couvrir les intervalles $[4 \times 10^{11}, 10^{13}], [2-4] \times 10^{13}, [6-8] \times 10^{13}$, et $[9-10] \times 10^{13}$ a été approximativement de 75 heures CPU pour générer les grands nombres premiers, et 225 heures CPU pour le crible. Ce dernier temps signifie que pendant le temps utilisé pour effectuer la partie crible, une moyenne de 3.2×10^8 mots de 64 bits par seconde CPU étaient "ou"-és. Le plus grand nombre de nombres premiers que nous ayons eu à calculer a été 413 : pour $e = 33\ 836\ 446\ 494\ 106$ et le premier $q_1 = 33\ 835\ 999\ 990\ 007$, il s'avéra que $e - q_i$ est composé pour $i = 1, \ldots, 412$, et premier pour $i = 413$, ($q_{413} = 33\ 836\ 000\ 002\ 499$ et $e - q_{413} = 446\ 491\ 607$).

4.2 *Expérimentations sur un serveur de calcul SGI avec 18 R8000 CPU*

L'algorithme implémenté sur la station de travail SGI est très proche de celui du Cray C916. Les nombres premiers jusqu'à 128×10^6 sont représentés dans un tableau binaire, que nous appelons ODD, d'un million d'entrées qui sont des longs entiers 64 bits : le j-ième bit du i-ième élément du tableau est égal à 1 si et seulement si $128i + 2j + 3$ est un nombre premier. De façon similaire, un autre tableau de la même taille, correspondant au tableau SIEVE de la section précédente, est utilisé pour noter les nombres composés : le j-ième bit du i-ième élément de ce second tableau est égal à 1 si et seulement si $128i + 2j + seed$ est décomposable en une somme de deux nombres premiers, où $seed$ dénote le nombre pair auquel la phase commence.

À ce point, la tâche du programme consiste à remplir toutes les entrées de SIEVE avec le plus grand mot de 64 bits i.e. $2^{64} - 1$. Le programme cherche la dernière entrée i pour laquelle la valeur de SIEVE$[i]$ n'est pas maximum et cherche le plus petit bit j de cette entrée qui n'est pas égal à 1. Alors, le nombre $128i+2j+seed$ n'a toujours pas été écrit comme une somme de deux nombres premiers. Le programme cherche alors la plus petite valeur k pour laquelle $128(i-k) + seed - 3$ et $128k + 2j + 3$ sont tous les deux premiers. Quand une telle valeur de k est trouvée, le tableau SIEVE commençant à l'entrée i peut être combiné avec le tableau ODD commençant à l'entrée k avec une opération "ou" comme précédemment. Le fait d'avoir un écart

de 128 dans la recherche des nombres premiers ne change pas la densité des nombres dont l'on s'attend à ce qu'ils soient premiers et a l'avantage d'éviter le décalage du tableau ODD. À la fin, pour gagner en efficacité, l'adressage dans le tableau SIEVE a été fait en utilisant une liste chaînée : cette liste contient seulement les valeurs i pour lesquelles SIEVE[i] n'est pas maximal. Alors après chaque opération "ou", la valeur résultante est comparée à $2^{64} - 1$ et s'il y a égalité, l'index correspondant est supprimé de la liste chaînée. Ainsi, la taille du tableau décroît au fur et à mesure du temps et globalement aucune opération "ou" inutile n'est faite. L'inconvénient est que l'adressage a été effectué par une redirection indirecte de pointeur et cela ralentit le programme au début de l'exécution. Les implémentations des versions avec et sans liste chaînée ont été testées sur une station DECSTATION 3100 avec des longueurs de mots variées et des tailles différentes des tableaux ODD et SIEVE. Le gain apporté par la version avec liste chaînée est apparu comme étant maximal pour des tableaux d'une longueur de 1.5×10^6 mots de 32 bits, avec un facteur de 1.59. Plus tard, quelques comparaisons ont été faites avec une version avec des entrées qui sont les nombres premiers jusqu'à 10^9. Le ratio entre les temps d'exécution était égal à 0.82 au bénéfice des dernières versions. D'autres améliorations n'ont pas été implémentées, e.g. anticipant les décompositions en blocs de nombres pairs suivant celui du tableau courant SIEVE, quand les indices sortent des limites de ce second tableau.

Des exécutions typiques consistent à vérifier 1350 intervalles consécutifs de nombres pairs de longueur 128.10^6 avec une exécution sur chacun des 18 processeurs R8000 de la station de travail SGI. Sept telles exécutions furent nécessaires pour traiter un intervalle de 2.10^{13}. Les intervalles qui ont été vérifiés sont $[10^{13}, 2.10^{13}]$, $[4.10^{13}, 6.10^{13}]$, et $[8.10^{13}, 9.10^{13}]$. Un nombre total de 324 exécutions a été nécessaire pour réaliser la tâche dans son intégralité. Les temps CPU utilisateur pour les différentes exécutions ont varié de 10 heures 33 mn pour l'exécution commençant à 9 158 401 000 000 et finissant à 9 331 201 000 000, jusqu'à 17 heures 12 mn pour l'exécution de 2 937 601 000 000 à 3 110 401 000 000. Le temps séquentiel total a été de 4083 heures 38 mn et le temps réel, qui est environ 18 fois plus petit, est environ 227 heures. Ces temps incluent la recherche des nombres premiers et le criblage. Le nombre de nombres premiers nécessités pour vérifier la décomposition de 64.10^6 entiers pairs consécutifs varie de 160 pour les intervalles commençant à 16 182 785 000 000 et 53 917 312 000 000, à 184 pour les intervalles commençant à 145 793 000 000. Quand on a vérifié sur les intervalles de longueur 10^9, le nombre moyen de nombres premiers a grossi jusqu'à 218.

5. Vérifier BGC près des hautes puissances de dix

En plus d'étendre G_2, nous avons également vérifié la conjecture binaire de Goldbach sur des intervalles de longueur 10^8 et 10^9 près des grandes puissances de dix. Le second auteur a vérifié les intervalles $[10^{5i}, 10^{5i} + 10^8]$, pour $i = 3, 4, \ldots, 20$, et le troisième auteur a vérifié les intervalles $[10^{10i}, 10^{10i} + 10^9]$, pour $i = 20, 21, \ldots, 30$.

5.1 Les intervalles $[10^{5i}, 10^{5i} + 10^8]$*, pour* $i = 3, 4, \ldots, 20$

Pour chaque intervalle $[B, B + 10^8]$, les 300 plus grands nombres premiers $\leq B$ ont été générés. Ici, les résultats de Jaeschke ne peuvent plus être utilisés parce que les nombres sont trop grands. À la place, nous avons d'abord généré les 300 plus grands nombres $\leq B$ qui sont passés à travers un test de pseudo-premier fort pour une base sélectionnée au hasard, et après nous avons prouvé la primalité de ces nombres avec un programme développé par H. Cohen, A.K. Lenstra, et D.T. Winter [5] : tous ces nombres s'avérèrent être premiers. Pour l'ensemble $\mathcal{P}$, nous avons pris les nombres premiers impairs inférieurs à $10^8 + 10^6$. La technique de crible était la même que celle utilisée sur le Cray C916 pour les nombres pairs jusqu'à 10^{14}.

Une sélection des résultats est donnée dans la Table 1. La seconde colonne donne la valeur de $(q_{300} - q_1) = (299 \log 10)$ qui serait proche de $\log_{10} B$, selon le théorème des nombres premiers. Elle illustre que le comportement local des nombres premiers peut dévier considérablement du comportement global connu. Le nombre moyen d'étapes nécessaire (sur les 18 intervalles considérés) était 217, avec une déviation standard de 23. Pour une distribution *uniforme* des bits dans le tableau ODD (plutôt que la distribution induite par les nombres premiers), le nombre moyen d'étapes a été 152, avec une déviation standard de 9. Cela s'accorde bien avec le nombre attendu d'étapes (214 dans le cas des nombres premiers et 150 dans le cas d'une distribution uniforme) mentionnée en Section 3.

5.2 Les intervalles $[10^{10i}, 10^{10i} + 10^9]$*, pour* $i = 20, 21, \ldots, 30$

À nouveau, le serveur de calcul SGI a été utilisé pour faire une implémentation similaire. Pour un intervalle de la forme $[B, B + 10^9]$, comme dans l'implémentation pour les décompositions jusqu'à 10^{14}, les nombres pairs étaient représentés par des bits dans un tableau de 7812500 mots de 64-bit. La technique du crible était la même que précédemment et utilisait aussi des listes chaînées. Pourtant, à cause de la taille des nombres, à nouveau les résultats de Jaeschke ne pouvaient pas être utilisés pour établir la primalité. Au lieu de ça, nous avons fait passer aux nombres candidats le test de pseudo-primalité de Miller-Rabin pour les bases 2, 3, 5 et 7 après un crible rapide d'essais de divisions. L'implémentation de cette phase a été faite en utilisant le système PARI. Dans une seconde phase, nous avons certifié la primalité de ces nombres par le programme de François Morain [1, 11] de preuve de la primalité par les courbes elliptiques. Sur un nœud R10000, les temps CPU pour la version C d'ECPP, que le troisième auteur avait à sa disposition variaient de 4 minutes pour des nombres à 200 chiffres décimaux à 60 minutes pour des nombres à 300 chiffres décimaux. Comme comparaison, la primalité de certains de ces nombres a été prouvée par le programme de Cohen, Lenstra et Winter [5] (pour des nombres jusqu'à 220 chiffres décimaux ; le temps CPU était de deux minutes par nombre sur une station de travail 180 MHZ IP32 SGI) et par le programme de Bosma et Van der Hulst [2] (pour les nombres ayant plus de 220 chiffres décimaux ; le temps moyen CPU était de sept minutes par nombre sur la même station de travail 180 MHZ IP32 SGI [3]). Le nombre de nombres premiers nécessité pour vérifier la BGC sur un intervalle de longueur 10^9 était en fait assez stable, variant de 222 à 231 pour les intervalles considérés avec une valeur moyenne de 225.
La Table 2 résume les résultats.

Table 1 Vérifier la conjecture de Goldbach sur les intervalles $[B, B + 10^8]$, pour $B = 10^{15}, 10^{20}, \ldots, 10^{100}$.

Notation

3. Le temps CPU nécessité par ce programme grossit avec la taille du nombre premier, mais de façon très erratique.

$[B, B+10^8]$: l'intervalle sur lequel la conjecture de Goldbach est vérifiée ;

$q_1, \ldots, q_{300}$: les plus grands 300 nombres premiers consécutifs $\leq B$, générés sur une station de travail SGI avec un processeur 100 MHZ IP22 ;

T_{pr} : la somme des temps-CPU en minutes utilisés pour générer les 300 plus grands nombres pseudo-premiers forts $< B$ (qui ont passé un test de pseudo-primalité sévère pour une base choisie au hasard) avec le package PARI et dont la primalité a été prouvée avec un code Fortran/C basé sur l'algorithme de preuve de primalité de Cohen-Lenstra (tous les pseudo-premiers "forts" se sont avérés être premiers) ;

N_1 : le plus petit entier positif tel que pour chaque nombre pair $e \in [B, B+10^8]$, il y a un indice i avec $1 \leq i \leq N_1$ tel que $e - q_i$ est un nombre premier ;

W : le "pire des cas" dans le test de la conjecture de Goldbach de l'intervalle $[B, B+10^8]$, i.e., $W - q_i$ est composé pour $i = 1, 2, \ldots, N_1 - 1$, mais premier pour $i = N_1$.

$\log_{10} B$	$\frac{q_{300}-q_1}{299 \log 10}$	$B - q_1$	$B - q_{300}$	T_{pr}	N_1	$W - B$
15	16.2	11159	11	1.9	243	87831838
20	19.8	13611	11	2.4	210	40249602
25	24.6	17063	123	3.0	240	91143618
30	31.9	21941	11	4.2	216	70421718
35	34.1	23477	23	5.7	182	84348372
40	37.2	25649	17	6.0	202	61919718
45	51.5	35469	9	8.0	283	80017866
50	45.3	31247	57	11	198	84955228
55	56.8	39183	111	16	218	88062574
60	58.9	40721	161	25	210	68370894
65	66.4	45951	269	32	193	80085838
70	72.6	50093	93	43	210	56324104
75	80.7	55779	191	53	224	31058458
80	82.6	56907	11	65	206	24403128
85	76.8	52919	27	82	203	45500944
90	95.6	65981	143	94	209	70588714
95	92.9	64011	53	122	207	88980634
100	99.0	68969	797	150	250	41229036

Table 2 Vérifier la conjecture de Goldbach sur les intervalles $[B, B+10^9]$, pour $B = 10^{200}, 10^{210}, \ldots, 10^{300}$.

Notation

$[B, B+10^9]$	:	l'intervalle sur lequel la conjecture de Goldbach est vérifiée ;
$q_1, q_2, \ldots$	:	la liste des nombres premiers nécessaire à la vérification de BGC sur $[B, B+10^9]$;
N_q	:	la cardinalité de l'ensemble précédent ;
T_{gen}	:	le temps nécessaire pour cribler l'intervalle et pour générer les N_q nombres pseudo-premiers forts avec le package PARI sur un seul nœud du SGI ;
T_{pp}	:	le temps séquentiel total utilisé par ECPP pour prouver la primalité des N_q pseudo-premiers (cette tâche a été en fait distribuée sur tous les nœuds du serveur de calcul) ;
N_{def}	:	le nombre de pseudo-premiers qui n'ont pas pu être certifiés par ECPP ;
W	:	le "pire des cas" pour la vérification de BGC dans l'intervalle, i.e. $W - q_i$ est composé pour $i = 1, 2, \ldots, N_q - 1$ mais premier pour $i = N_q$.

$\log_{10} B$	N_q	$\frac{qN_q - q_1}{64.N_q.\log 10}$	$B - q_1$	$qN_q - B$	$W - B$
200	224	30281.7	97283	999497853	999786382
210	222	30559.2	243203	999503869	999686796
220	222	30557.9	177539	999530109	999620578
230	228	29754.9	112643	999634941	999983752
240	228	29763.3	191747	999836541	999872854
250	231	29381.5	701699	999488509	999991806
260	226	30026.2	174467	999837181	999864924
270	223	30418.1	112643	999503997	999697006
280	225	30151.7	32003	999714813	999837064
290	226	30023.9	115331	999819517	999872646
300	227	29885.3	32771	999692157	999821434

$\log_{10} B$	T_{gen}	T_{pp}	N_{def}
200	33 mn 52 s	14 h 20 mn 02 s	4
210	35 mn 56 s	39 h 20 mn 31 s	5
220	44 mn 47 s	48 h 45 mn 05 s	21
230	48 mn 43 s	62 h 57 mn 39 s	25
240	55 mn 32 s	72 h 20 mn 47 s	23
250	1 h 11 mn 38 s	91 h 10 mn 43 s	19
260	1 h 16 mn 42 s	104 h 25 mn 31 s	17
270	1 h 13 mn 02 s	120 h 35 mn 28 s	15
280	1 h 45 mn 09 s	98 h 31 mn 15 s	27
290	1 h 39 mn 24 s	177 h 37 mn 48 s	17
300	2 h 04 mn 44 s	219 h 54 mn 59 s	41

Bibliographie

[1] A.O.L. Atkin and F. Morain. *Elliptic curves and primality proving*, Mathematics of Computation, 61 :29-68, 1993.

[2] W. Bosma and M.-P. van der Hulst. *Primality proving with cyclotom*, PhD thesis, University of Amsterdam, December 1990.

[3] J.R. Chen and T.Z. Wang, *On the odd Goldbach problem*, Acta Math. Sinica **32** (1989), p. 702-718 (in Chinese).

[4] J.R. Chen and T.Z. Wang, *On odd Goldbach problem under General Riemann Hypothesis*, Science in China **36** (1993), p. 682-691.

[5] H. Cohen and A.K. Lenstra, *Implementation of a new primality test*, Math. Comp. **48** (1987), p. 103-121.

[6] J-M. Deshouillers, G. Effinger, H. te Riele and D. Zinoviev, *A complete Vinogradov 3-primes theorem under the Riemann hypothesis*, Electronic Research Announcements of the AMS **3** (1997), p. 99-104 (September 17, 1997). http://www.ams.org/journals/era/home-1997.html.

[7] A. Granville, J. van de Lune and H.J.J. te Riele, *Checking the Goldbach conjecture on a vector computer*, Number Theory and Applications (R.A. Mollin, ed.), Kluwer, Dordrecht, 1989, p. 423-433.

[8] G.H. Hardy and L.E. Littlewood, *Some problems of 'Partitio Numerorum'; III : On the expression of a number as a sum of primes*, Acta Math. **44** (1922/3), p. 1-70.

[9] G. Jaeschke, *On strong pseudoprimes to several bases*, Math. Comp. **61** (1993), p. 915-926.

[10] L. Kaniecki, *On Šnirelman's constant under the Riemann hypothesis*, Acta. Arithm. **72** (1995), p. 361-374.

[11] F. Morain. *Courbes Elliptiques et Tests de Primalité*, PhD thesis, L'Université Claude Bernard, Lyon I, September 1990. Introduction in French, body in English.

[12] O. Ramaré, *On Šnirelman's Constant*, Ann. Scuola Norm. Sup. Pisa **22** (1995), p. 645-706.

[13] Y. Saouter, *Checking the odd Goldbach conjecture up to* 10^{20}, Math. Comp., **67** (1998), p. 863-866.

[14] L. Schoenfeld, *Sharper Bounds for the Chebyshev Functions* $\theta(x)$ *and* $\psi(x)$ II, Math. Comp. **30** (1976), p. 337-360.

[15] M.-K. Shen, *On Checking the Goldbach conjecture*, BIT **4** (1964), p. 243-245.

[16] M.K. Sinisalo, *Checking the Goldbach conjecture up to* 4.10^{11}, Math. Comp. **61** (1993), p. 931-934.

[17] M.L. Stein and P.R. Stein, *Experimental results on additive 2 bases*, Math. Comp. **19** (1965), p. 427-434.

[18] I.M. Vinogradov, *Representation of an odd number as a sum of three primes*, Comptes Rendus (Doklady) de l'Académie des Sciences de l'URSS, **15** (1937), p. 291-294.

[19] D. Zinoviev, *On Vinogradov's constant in Goldbach's ternary problem*, J. Number Th. **65** (1997), p. 334-358.